

# Neural Color Operators for Sequential Image Retouching

Yili Wang<sup>1</sup>\*, Xin Li<sup>2</sup>, Kun Xu<sup>1</sup>\*\*, Dongliang He<sup>2</sup>, Qi Zhang<sup>2</sup>,  
Fu Li<sup>2</sup>, and Errui Ding<sup>2</sup>†

<sup>1</sup> BNRist, Department of CS&T, Tsinghua University

<sup>2</sup> Department of Computer Vision Technology (VIS), Baidu Inc.

yiliw.thu@gmail.com, lixin41@baidu.com, xukun@tsinghua.edu.cn,  
{hedongliang01, zhangqi44, lifu, dingerrui}@baidu.com

**Abstract.** We propose a novel image retouching method by modeling the retouching process as performing a sequence of newly introduced trainable *neural color operators*. The neural color operator mimics the behavior of traditional color operators and learns pixelwise color transformation while its strength is controlled by a scalar. To reflect the homomorphism property of color operators, we employ equivariant mapping and adopt an encoder-decoder structure which maps the non-linear color transformation to a much simpler transformation (i.e., translation) in a high dimensional space. The scalar strength of each neural color operator is predicted using CNN based strength predictors by analyzing global image statistics. Overall, our method is rather lightweight and offers flexible controls. Experiments and user studies on public datasets show that our method consistently achieves the best results compared with SOTA methods in both quantitative measures and visual qualities. Code is available at <https://github.com/amberwangyili/neurop>.

**Keywords:** image retouching, image enhancement, color operator, neural color operator

## 1 Introduction

In the digital eras, images are indispensable components for information sharing and daily communication, which makes improving the perceptual quality of images quite demanding in many scenarios. While professional retouching software such as Adobe Photoshop and Lightroom provide various retouching operations, the effectiveness and proficiency of using them still require expertise on the part of the users. Automation of such tedious editing works is highly desirable and has been extensively studied for decades.

Recently, deep learning has exhibited stunning success in many computer vision and image processing tasks [34,20], which generally uses networks to learn sophisticated mappings from paired data. In the pioneering work by Bychkovsky

---

\* Work done during Yili Wang’s internship at VIS, Baidu.

\*\* Kun Xu is the corresponding author.

*et al.* [6], a large scale of input and expert-retouched image pairs have been collected: the constructed MIT-Adobe FiveK dataset has enabled high-quality supervised learning and drives the mainstream towards data-driven methods.

Among these methods, *sequential image retouching* [39,16,38,45] is rather attractive. Following human’s step-by-step workflow, they model the image retouching process as a sequence of standard color operators (*i.e.*, brightness or contrast adjustments), each of which is controlled by a scalar parameter. Besides generating a retouched image, they also produce a semantically meaningful history of retouching operators. With the retouching history on hand, the process is more interpretable, offers convenient controls for re-editing, and could be used in applications like photo manipulation tutorial generation [12] and image revision control [8]. However, due to the limited expressiveness of standard color operators, these methods require a relatively large number of operators to reproduce the complex retouching effects, which makes it less robust to accurately predict the operator parameters and hard to achieve high-fidelity retouched results (*i.e.*, high PSNR scores) compared to other state-of-the-art retouching methods [49,14,33,50].

To address this issue, we introduce a novel trainable color operator, named *neural color operator*. It is designed to mimic the behavior of traditional color operators, which maps a 3D RGB color to a modified color according to a scalar indicating the operator’s strength. Instead of using a fixed function as in standard color operators, we allow the parameters of a neural color operator to be learned in order to model more general and complex color transformations. To reflect the *homomorphism property* of color operators (*i.e.*, adjusting exposure by +2, followed a second +3 adjustment, is approximately equivalent to a single operator that adjusts exposure by +5), we employ *equivariant mapping* and adopt an encoder-decoder structure, which maps the color transformation in 3D RGB space to a much simpler transformation (*i.e.*, translation) in a high dimensional feature space where the scalar strength controls the amount of translation.

We further propose an automatic, lightweight yet effective sequential image retouching method. We model the retouching process as applying a sequence of pixel-wise neural color operators to the input image. The strength of each neural color operator is automatically determined by the output of a CNN based *strength predictor* using global statistics of deep image features computed from downsampled intermediate adjusted images. The neural color operators and strength predictors are jointly trained in an end-to-end manner, with a carefully designed initialization scheme.

**Contributions.** Our contributions are summarized as follows: First, we introduce neural color operator — a novel, trainable yet interpretable color operator. It mimics the behavior of color operators whose strength is controlled by a scalar, and could effectively model sophisticated global color transformations. Second, based on neural color operators, we propose a lightweight and efficient method with only 28k parameters for sequential image retouching. Without any bells and whistles, experiments and user studies show that our method consistently achieves the best performance on two public datasets, including MIT-Adobe

FiveK [6] and PPR10K [30], compared with state-of-the-art methods from both qualitative measures and visual qualities. Furthermore, our method inherits the nice properties of sequential image retouching, which allows flexible controls and convenient re-editing on results by intuitively adjusting the predicted scalar strengths using three sliders in real-time.

## 2 Related works

Computational methods for automatic image retouching have kept evolving over the years — from traditional heuristics such as spatial filters [4], to histogram-based methods [3,26,28,40,44], and recent learning-based approaches. Here we give a general review on this most recent branch of researches.

**Image-to-Image Translation based Methods.** These methods generally consider image retouching as a special case of image-to-image translation and use neural networks to directly generate the enhanced images from input images [18,19,24,9,37,21,48,50]. While these methods can produce visually pleasing results, they have some common limitations: they usually employ large networks and consume too much computational resources when applied to high-resolution images; besides, the use of downsampled convolutional layers may easily lead to artifacts in high frequency textures. Some methods have been proposed to alleviate these issues by decomposition of reflectance and illumination [47,42], representative color transform [25], or Laplacian pyramid decomposition [1,31].

**Sequential Image Retouching.** This category of methods [39,16,38,45] follow the step-by-step retouching workflow of human experts, which models the process as a sequence of several color operators where the strength of each operator is controlled by a scalar. This modeling paradigm is quite challenging because it requires more supervision of editing sequences. Some researchers [16,38] resort to deep reinforcement learning algorithms which are known to be highly sensitive to their numerous hyper-parameters [15]. Shi *et al.* [39] propose an operation-planning algorithm to generate pseudo ground-truth sequences, hence making the network easier to train. However, it needs extra input text as guidance. Overall, these methods provide an understandable editing process, and allow convenient further controls on results through intuitive adjustments of the color operators. However, due to the limited expressiveness of standard color operators, they require a sequence with a relatively large number of operators to reproduce complex retouching effects, which is harder to learn and leads to less satisfactory performance (*i.e.*, lower PSNR scores).

**Color Transformation based Methods.** These methods usually use a low-resolution image to extract features and predict the parameters of some predefined global or local color transformation, and later apply the predicted color transformation to the original high-resolution image. Different color transformations have been used in existing works, including quadratic transforms [46,7,32,41],

local affine transforms [11], curve based transforms [6,13,29,23,36], filters [10,35], lookup tables [49,43], and customized transforms [5]. Compared with image-to-image translation based methods, these methods usually use smaller models and are more efficient. However, their capabilities are constrained by the predefined color transformation and might be inadequate to approximate highly non-linear color mappings between low and high quality images.

Recently, He *et al.* [14,33] proposed Conditional Sequential Retouching Network (CSRNet), which models the global color transformation using a 3-layer multi-layer perceptron (MLP). The MLP is influenced through global feature modulation via a 32D conditional vector extracted from the input image through a conditional network. Their method is lightweight and achieves nice performance. However, the 32D condition vector is hard to interpret and control. While CSRNet is named with ‘sequential’, we prefer to regard it as a color transformation based method instead of a sequential image retouching method. This is because it does not explicitly model retouching operators or generate a meaningful retouching sequence, nor does it produce intermediate adjusted images, all of which are important features of sequential image retouching.

**Our motivation.** We are motivated by the seemingly intractable dilemma in the context of previous retouching methods: while “black-boxing” the standard operators [14,33] can improve model expressiveness such non-interpretable parameterization leads to the lack of controllability; though good interpretability can be achieved by using sequential methods with standard post-processing operators [39,16,38,45], these methods require a relatively large number of operators to reproduce the complex retouching effects but also make it less robust to accurately predict the parameters and fail to achieve satisfactory results.

Specifically, we seek to resolve this dilemma by disentangling the complex nonlinear retouching process into a series of easier transformations modeled by our trainable neural color operators with interpretable controls, while still maintaining high efficacy and lightweight structures.

### 3 Neural Color Operator

**Definition.** A neural color operator (short as neural operator or neurOp) mimics the behavior of traditional global color operators. It is defined to learn a pixel-wise global color mapping  $\mathcal{R}$  from an input 3D RGB color  $\mathbf{p}$ , with a 1D scalar  $v$  indicating the operator’s strength, to an output 3D RGB color  $\mathbf{p}'$ :

$$\mathbf{p}' = \mathcal{R}(\mathbf{p}, v). \quad (1)$$

For simplicity, we restrict the scalar strength  $v$  to be normalized in  $[-1, 1]$ .

**Color Operator Properties.** By carefully studying standard global color operators, such as **exposure**, **black clipping**, **vibrance** in *Lightroom*, and **contrast**, **brightness** in *Snapseed*, and general hue, saturation and gamma adjustments, we find those operators usually satisfy the following properties:

- 1. The effect of a color operator is controlled in a continuous way by adjusting the scalar strength.
- 2. The color value should keep unchanged if strength is zero, *i.e.*,  $\mathbf{p} = \mathcal{R}(\mathbf{p}, 0)$ .
- 3. The larger the strength is, the larger the color changes. For example, adjusting brightness by +2 should incur more changes than +1 adjustment.
- 4. From a mathematical aspect, the operator should approximately satisfy the *homomorphism property*, *i.e.*,  $\mathcal{R}(\mathbf{p}, v_1 + v_2) \approx \mathcal{R}(\mathcal{R}(\mathbf{p}, v_1), v_2)$ . For example, adjusting exposure by +3, followed by a second operator that adjusts exposure by +2, is approximately equivalent to a single operator that adjusts exposure by +5.

Recall that our goal is to mimic the behavior of traditional color operators. Hence, The above color operator properties should be considered in the design of our neural color operators. Our main observation is: since the translation function naturally satisfies those properties, *i.e.*, for  $f(x, v) = x + v$  we easily have  $f(x, v_1 + v_2) = f(f(x, v_1), v_2)$ , if we could map the non-linear color transformation in the 3D RGB space to a simple translation function in a high-dimensional feature space, the neural color operators would also satisfy those properties.

**Network Structure.** Based on the observation, we employ *equivariant mapping*, which has been widely used in other applications like geometric deep learning [22,27], for the purpose. Specifically, we adopt an encoder-decoder structure, as shown in Figure 1 (top right). The encoder  $E$  transforms the input 3D RGB color  $\mathbf{p}$  to a high-dimensional (*i.e.*, 64D) feature vector  $\mathbf{z}$ , then we perform a simple translation in the feature space  $\mathbf{z}' = \mathbf{z} + v \cdot \mathbf{i}$ , where the translation direction  $\mathbf{i}$  is simply set as an all-one vector in the feature space and  $v$  controls the distance of translation. Finally, the decoder  $D$  transforms the feature vector  $\mathbf{z}'$  back to the 3D RGB space to obtain the output color  $\mathbf{p}'$ . In general, we define the mapping of a neural color operator  $\mathcal{R}$  as:

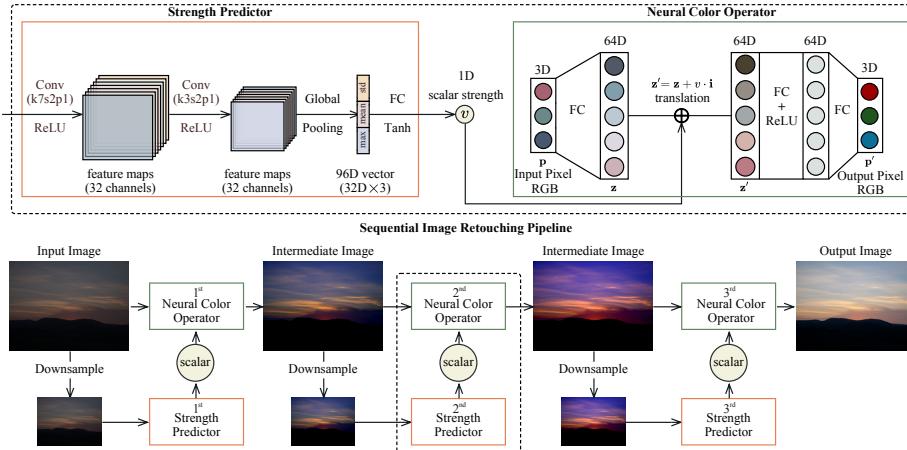
$$\mathbf{p}' = \mathcal{R}(\mathbf{p}, v) = D(E(\mathbf{p}) + v \cdot \mathbf{i}). \quad (2)$$

To maintain a lightweight structure, the encoder contains only one fully connected (FC) layer (without any hidden layers), and the decoder contains two FC layers where the first layer uses ReLU activations. Different from traditional encoders which usually map data to a lower dimensional latent space, our encoder maps colors to a higher dimensional feature space. We provide visualization of the 64D feature vectors in the supplemental document.

Note that our neural color operator can only approximately (but not strictly) satisfy the above color operator properties, since the decoder is not theoretically guaranteed to be the exact inverse of the encoder, *i.e.*,  $D(E(\mathbf{p})) \approx \mathbf{p}$ .

## 4 Automatic Sequential Image Retouching

In this section, we present how to develop an automatic, lightweight yet effective sequential image retouching method based on our proposed neurOps.



**Fig. 1.** Pipeline of our approach and network structures. Top left: structure of a strength predictor. Parameters of two convolutional layers are shared among all strength predictors ( $k7s2p1$  denotes a kernel size  $7 \times 7$ , a stride of 2 and a padding of 1). Top right: network structure of a neural color operator. Parameters are not shared among different neural color operators. Bottom: the overall pipeline of our method.

#### 4.1 Problem Setup

Given an input image  $I \in \mathbb{R}^{H \times W \times C}$  with an arbitrary resolution, our goal is to generate a retouched image  $I^R \in \mathbb{R}^{H \times W \times C}$  by sequentially applying  $K$  pixel-wise color transformations modeled by neurOps:

$$I_k = \mathcal{R}_k(I_{k-1}, v_k), \quad 1 \leq k \leq K, \quad (3)$$

where  $I_0 = I$  is the input image,  $I_k$  ( $1 \leq k \leq K - 1$ ) are intermediate images, and  $I^R = I_K$  is the output image.  $\mathcal{R}_k$  is the  $k$ -th neurOp, and  $v_k$  is a scalar that controls its strength. A neurOp maps each pixel's input color to an output color in a pixel-wise manner. In our experiments, we use  $K = 3$  neurOps for the best trade-off between model size and efficacy.  $H$ ,  $W$  and  $C$  denote the height, width, and channel size of the image, respectively.

To make the retouching process automatic, we predict the strength of each neurOp by strength predictors:

$$v_k = \mathcal{P}_k(I_{k-1}^\downarrow), \quad 1 \leq k \leq K, \quad (4)$$

where  $\mathcal{P}_k$  is the  $k$ -th strength predictor, and  $I_{k-1}^\downarrow$  is the bilinearly downsampled image from intermediate image  $I_{k-1}$  by shortening the longer image edge to 256. We use intermediate images instead of always using the original input image  $I$  to infer the strength parameter. This is motivated by human's editing workflow: an artist also needs intermediate visual feedbacks to decide the next editing step [16]. Figure 1 (bottom) illustrates the overall pipeline of our approach. Note that we do not share parameters between the 3 neurOps.

## 4.2 Strength Predictor

A strength predictor is used to predict the scalar strength used in a neurOp. Global image information is required for such prediction, i.e., an exposure adjustment needs the average intensity of the image. Hence, we use a downsampled intermediate edited image from the previous editing step as the input to the strength predictor.

The strength predictor contains 4 layers: two convolutional layers, a global pooling layer, and a FC layer. First, we use the first two convolutional layers to extract 32-channel feature maps. Then, we apply global pooling to obtain a 96D vector. Specifically, we employ three different pooling functions to compute channel-wise maximum, average, and standard deviation, respectively, and concatenate the three 32D vectors together. Finally, the FC layer, which acts as a predicting head, maps the 96D vector to a scalar strength in  $[-1, 1]$ . The top left of Figure 1 shows the network structure of a strength predictor.

To reduce the overall size of our network, we force the parameters of the two convolutional layers to be shared among all 3 strength predictors. The parameters of the predicting head (*i.e.*, the last FC layer) are not shared.

Our strength predictor is inspired by the conditional network in CSRNet [33,14] in order to maintain a lightweight structure. It is improved in several ways. First, we take intermediate images instead of the original image as input to better reflect human’s editing workflow. Second, their conditional network outputs a 32D vector while our strength predictor generates a scalar. Third, we use three pooling functions instead of only one average pooling function which are demonstrated to be more helpful.

## 4.3 Loss Function and Training

We jointly train our neurOps and strength predictors by optimizing a predefined loss function in an end-to-end fashion. In practice, we find that a carefully designed initialization for the neurOps is also rather helpful.

**Loss Function.** For a single training image  $I$ , denoting its ground truth retouched image as  $I^{GT}$  and the predicted retouched image as  $I^R$ , respectively, the loss  $\mathcal{L}$  is defined as the weighted sum of a reconstruction loss  $\mathcal{L}_r$ , a total variation (TV) loss  $\mathcal{L}_{tv}$ , and a color loss  $\mathcal{L}_c$ :

$$\mathcal{L} = \mathcal{L}_r + \lambda_1 \mathcal{L}_{tv} + \lambda_2 \mathcal{L}_c, \quad (5)$$

where  $\lambda_1$  and  $\lambda_2$  are two balancing weights empirically set as  $\lambda_1 = \lambda_2 = 0.1$ . The reconstruction loss measures L1 difference between the predicted image and the ground truth:

$$\mathcal{L}_r = \frac{1}{CHW} \|I^R - I^{GT}\|_1. \quad (6)$$

A TV loss [2] is included to encourage spatially coherent predicted images:

$$\mathcal{L}_{tv} = \frac{1}{CHW} \|\nabla I^R\|_2, \quad (7)$$

where  $\nabla(\cdot)$  denotes the gradient operator. Besides, we include a color loss [32,42], which regards RGB colors as 3D vectors and measures their angular differences. Specifically, the color loss is defined as:

$$\mathcal{L}_c = 1 - \frac{1}{HW} \angle(I^R, I^{GT}), \quad (8)$$

where  $\angle(\cdot)$  is an operator that computes the pixel-wise average of the cosine of the angular differences. The overall loss is computed as the sum of losses (Equation 5) over all training images.

**Initialization Scheme for Neural Color Operators.** Instead of initializing the neurOps from scratch, we find that initializing them with standard color operators is a better choice. By analyzing the retouching histories in the MIT-Adobe FiveK dataset [6], we picked up the 3 most commonly used standard operators in the dataset, which are `black clipping`, `exposure`, and `vibrance` in *Lightroom*. For initialization, we intend to let our 3 neurOps reproduce the above 3 standard operators, respectively.

Since the formulas of those Lightroom operators are not publicly available, we have selected a small number of images (*i.e.*, 500 images) from the training set of MIT-Adobe FiveK, and manually retouched those images using those standard operators. Specifically, for each standard operator, for each input image  $I$ , we manually generate  $M$  (*i.e.*,  $M = 40$ ) retouched images (*i.e.*,  $\{I_m\}$ ,  $1 \leq m \leq M$ ) with uniformly distributed levels of strengths. The strength used for each retouched image  $I_m$  is denoted as  $v_m$ .

Then, we enforce each neurOp  $\mathcal{R}$  to reproduce the manually retouched images from each standard operator as much as possible. To do so, we optimize each neurOp by minimizing the sum of a unary loss and a pairwise loss over all selected images. The unary loss  $\mathcal{L}_1$  enforces that the color keeps unchanged when the strength is zero:

$$\mathcal{L}_1 = \frac{1}{M} \sum_m \|\mathcal{R}(I_m, 0) - I_m\|_1. \quad (9)$$

The pairwise loss  $\mathcal{L}_2$  enforces that the neurOp behaves the same as the given standard operator for specific strength values:

$$\mathcal{L}_2 = \frac{1}{M(M-1)} \sum_{m \neq n} \|\mathcal{R}(I_m, v_n - v_m) - I_n\|_1. \quad (10)$$

For optimization, we use the Adam optimizer with an initial learning rate of  $5e^{-5}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.99$ , and a mini-batch size of 1, and run 100,000 iterations.

**Joint Training.** After initializing neurOps using the scheme described above, and initializing the strength predictors with random parameters, we jointly train our neurOps and strength predictors in an end-to-end fashion towards minimizing the overall loss function (Equation 5). We also use the Adam optimizer with the same configurations as in the initialization scheme, and run 600,000 iterations. For data augmentation, we randomly crop images and then rotate them by multiples of 90 degrees.

## 5 Experiments

**Datasets.** The MIT-Adobe FiveK dataset [6] has been widely used for global image retouching and enhancement tasks. It consists of 5000 images in RAW format and an Adobe Lightroom Catalog that contains the adjusted rendition setting by five experts. We follow the common practice and use rendition version created by expert C as the ground truth. There are two public variations in terms of input rendition setting. The first variation is provided by Hu *et al.* [16], which choose the input `with Daylight WhiteBalance minus 1.5` and export them to 16 bits ProPhotoRGB TIFF images. We refer to it as *MIT-Adobe-5K-Dark*. The second variation is provided by Hwang *et al.* [17] which adopt the input `As Shot Zeroed` and export them to 8 bits sRGB JPEG images. We refer to it as *MIT-Adobe-5K-Lite*.

PPR10K [30] is a recently released image retouching dataset. It contains more than 11,000 high-quality portrait images. For each image, a human region mask is given and three retouched images by three experts are provided as ground truth. We refer to the three retouched variations as *PPR10K-a*, *PPR10K-b*, and *PPR10K-c*, respectively.

We train and evaluate our method on all 5 variations of both datasets. We follow the same split of training and testing sets as in previous works. For the MIT-Adobe FiveK dataset, all images are resized by shortening the longer edge to 500. For the PPR10K dataset, we use image resolution of 360p. All the experiments are performed on a PC with an NVIDIA RTX2080 Ti GPU. It takes about 2 hours for initialization, and takes about 9 and 15 hours for training on MIT-Adobe FiveK and PPR10K, respectively.

### 5.1 Comparison and Results

**Comparison.** We compare our method with a series of state-of-the-art methods, including White-Box [16], Distort-and-Recover [38], HDRNet [11], DUPE [42], MIRNet [48], Pix2Pix [20], 3D-LUT [49], CSRNet [14,33] on *MIT-Adobe-5K-Dark*, and DeepLPF [35], IRN [50] on *MIT-Adobe-5K-Lite*, and CSRNet, 3D-LUT, 3D-LUT+HRP [30] on *PPR10K*.

For the PPR10K dataset, since each image is associated with a human region mask, we also provide a variant of our method that additionally considers human region priority (HRP) [30] in training, which we refer to as NeurOp+HRP. This is done by slightly modifying the L1 loss in Equation 6 to be a weighted one, *i.e.*, pixels inside human region have larger weights (*e.g.*, 5) while other pixels have smaller weights (*e.g.*, 1).

As shown in Table 1, our method consistently achieves the best performance on all dataset variations, demonstrating the robustness of our method. In contrast, while CSRNet performs relatively well on MIT-Adobe-5K-Dark, it performs less satisfactory on PPR10K, *i.e.*, their PSNR is lower than ours by 2db on PPR10K-a, which is consistent with the benchmark carried out in [30]. Furthermore, our method is the most lightweight one (*i.e.*, only 28k parameters).

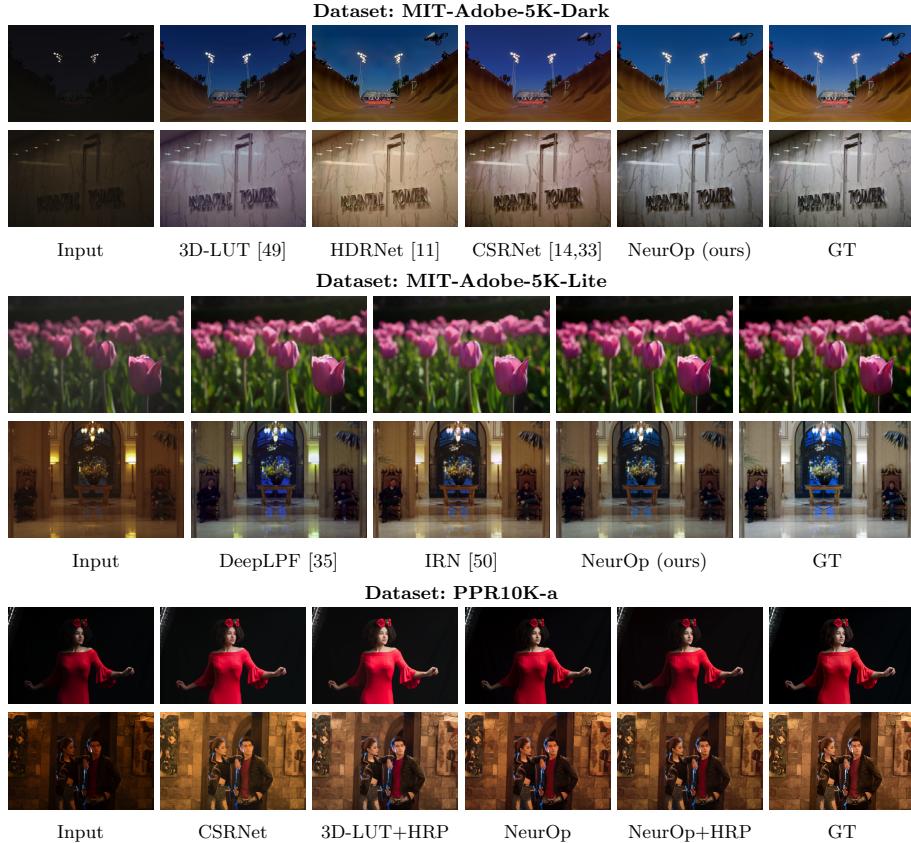
**Table 1.** Quantitative comparison with state-of-the-art methods. The best results are boldface and the second best ones are underlined.

Dataset	Method	PSNR↑	SSIM↑	$\Delta E_{ab}^{*\downarrow}$	#params	Dataset	Method	PSNR↑	SSIM↑	$\Delta E_{ab}^{*\downarrow}$
MIT- -Adobe -5K-Dark	White-Box [16]	18.59	0.797	17.42	8,561,762	10K-a	CSRNet [14,33]	24.24	0.937	9.75
	Dis.& Rec. [38]	19.54	0.800	15.44	<u>259,263,320</u>		3D-LUT [49]	25.64	-	-
	HDRNet [11]	22.65	0.880	11.83	482,080		3D-LUT+HRP [30]	25.99	0.952	8.95
	DUPE [42]	20.22	0.829	16.63	998,816		NeurOp (ours)	<u>26.32</u>	<u>0.953</u>	<u>8.81</u>
	MIRNet [48]	19.37	0.806	16.51	31,787,419		NeurOp+HRP (ours)	<b>26.46</b>	<b>0.955</b>	<b>8.80</b>
	Pix2Pix [20]	21.41	0.749	13.26	11,383,427	10K-b	CSRNet [14,33]	23.93	0.938	9.83
	3D-LUT [49]	23.12	0.874	11.26	593,516		3D-LUT [49]	24.70	-	-
	CSRNet [14,33]	<u>23.86</u>	<u>0.897</u>	<u>10.57</u>	<u>36,489</u>		3D-LUT+HRP [30]	25.06	0.945	9.36
	NeurOp (ours)	<b>24.32</b>	<b>0.907</b>	<b>10.10</b>	<b>28,108</b>		NeurOp (ours)	<u>25.45</u>	<u>0.946</u>	<u>9.21</u>
MIT- Adobe -5K-Lite	DeepLPF [35]	23.63	0.875	10.55	<u>1,769,347</u>		NeurOp+HRP (ours)	<b>25.82</b>	<b>0.951</b>	<b>8.97</b>
	IRN [50]	<u>24.27</u>	<u>0.900</u>	<u>10.16</u>	11,650,752	10K-c	CSRNet [14,33]	24.35	0.929	9.92
	NeurOp (ours)	<b>25.09</b>	<b>0.911</b>	<b>9.93</b>	<b>28,108</b>		3D-LUT [49]	25.18	-	-

Figure 2 shows visual comparisons. 3D-LUT sometimes generates color banding artifacts due to the use of color space interpolation. Other methods easily lead to color shifting problems especially when the input image has a very low exposure or has a very different temperature. Generally, the results of our method have fewer artifacts and are closer to the ground truth. More visual comparisons are given in the supplemental document.

**User Study.** We have conducted a user study to evaluate the subjective visual quality of our method on both dataset variations of MIT-Adobe-5K. For MIT-Adobe-5K-Dark, we compare with CSRNet [14,33], while for MIT-Adobe-5K-Lite, we choose IRN [50] for comparison, since the two methods achieve the second best performance on the two variations, respectively. For each dataset variation, we randomly select 50 images from the testing set and invite 10 participants (totally 20 participants are invited). For each selected image, for each participant, three retouched images are displayed: the ground truth image, and the two images generated by our method and the competing method (the latter two are displayed in random order). The participant is asked to vote which one of the latter two is visually more pleasing and more similar to the ground truth. Figure 3 (a) shows the results of user study, which suggest that our retouched images are visually more appreciated than those of competing approaches.

**Controllability.** While our method could generate the retouched images in a fully automatically way, we still allow users to intuitively adjust the results by changing the predicted scalar strengths using three sliders in real-time. Figure 3 (b) shows some examples. We find that adjusting the strengths could result in meaningful edits. For example, increasing the strength of the first neurOp amplifies the shadow and adds more contrast. Increasing the second tends to give more brightness to the middle tone while preserving the highlight proportion unchanged. Adjusting the third makes the image look cooler or warmer. More

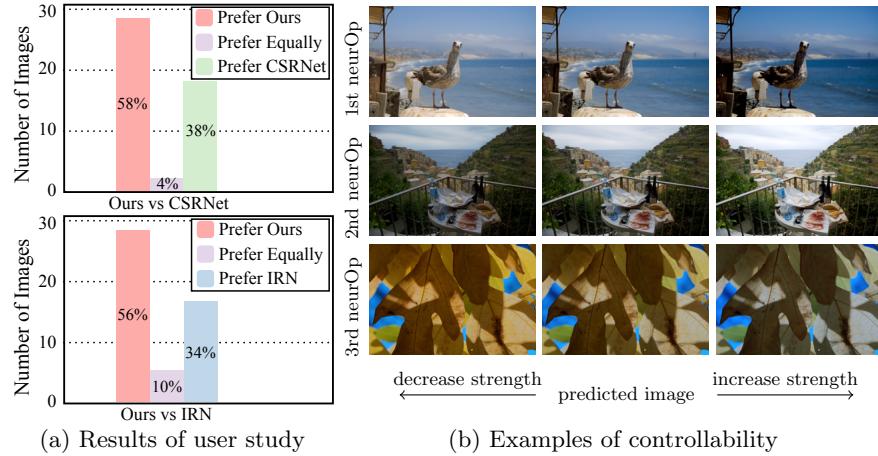


**Fig. 2.** Visual comparison with state-of-the-art methods.

examples of controllability as well as the intermediate images are provided in the supplemental document.

Existing methods such as CSRNet [14,33] achieve controlling through a linear interpolation between the retouched image and the original image by adjusting the interpolation weight. However, such controllability is rather limited. First, linear interpolation cannot faithfully reproduce sophisticated and highly non-linear color transformations. Second, only one degree of freedom is usually not enough for detailed color adjustments.

**Timing and Memory Consumption.** Our method runs in real-time. It takes 4 ms for a  $500 \times 333$  image or 19 ms for an image with 1M pixels, which is slightly slower than CSRNet and 3D-LUT. However, it is worthy since our method has smaller model size (*i.e.*, only 28k parameters), achieves better performance (*i.e.*, higher PSNR scores), and inherits advantages of sequential image retouching which they do not possess.



**Fig. 3.** User study and controllability. (a) Results of user study. (b) Examples of controllability. The middle column shows our automatically retouched images. The left and right columns show the results by further adjusting strengths of specific neurOps.

**Table 2.** Ablation study for neural color operators. (a) Different initialization schemes. (b) Initialization with different ordering of standard operators.  $\overrightarrow{vbe}$  denotes using the order of **vibrance**, **black clipping**, and **exposure** for initialization. (c) Different number of neurOps. (d) Our method (final model).

Configs	(a) Initialization		(b) Order			(c) Number		(d) Ours			
	random	standard fix	$\overrightarrow{vbe}$	$\overrightarrow{veb}$	$\overrightarrow{evb}$	$\overrightarrow{ebv}$	$\overrightarrow{bve}$	$K=1$	$K=2$	$K=4$	$\xrightarrow{\text{finetune, bev, } K=3}$
PSNR $\uparrow$	22.61	23.78	23.93	24.17	24.11	24.12	24.23	21.97	23.5	24.17	24.32

For memory consumption, our strength predictor takes less than 5M, our neroOp takes less than 1K per pixel. Overall, by processing pixels in batches (*i.e.*, 4096 pixels a batch), our method takes less than 10M memory for input images with arbitrary resolutions (storage for input/output images not included).

**Benefits of Our Method.** Due to the nice properties of neural color operators, the advantages of our method are multi-fold. First, it is rather lightweight and runs in real-time, hence, it could be easily deployed in mobile devices. Second, it inherits the advantages of sequential image retouching methods: providing an understandable editing history and offering convenient and flexible controls. Last but not least, besides the above advantages, our method still consistently achieves the best performance compared to state-of-the-art methods on two public datasets in both quantitative measures and visual qualities, as demonstrated by experiments and user studies.

**Table 3.** Ablation study for strength predictors and loss function. (a) Different pooling configurations in strength predictors. (b) Other design components in strength predictors. (c) Different configurations for the loss function. (d) Our method (final model).

Configs	(a) Pooling Layer			(b) Strength Predictor		(c) Loss Function			(d) Ours
	aver	aver+max	aver+std	ori-img-input	non-share-layer	$\mathcal{L}_r$	$\mathcal{L}_r + \mathcal{L}_c$	$\mathcal{L}_r + \mathcal{L}_{tv}$	
PSNR↑	23.96	24.27	24.11	23.91	24.35	24.22	24.30	24.26	24.32
#Params	27,916	28,012	28,012	28,108	56,076	28,108	28,108	28,108	28,108

## 5.2 Ablation Study

We conduct a series of ablation studies to justify our training strategy as well as the design choices of our neurOps and strength predictors. Except for the ablated parts, we retrain the ablated models adopting the same setting as our final models. We make quantitative comparisons using average PSNR achieved on the testing images of MIT-Adobe-5K-Dark as it is generally more challenging.

**Neural Color Operators.** We first verify the effectiveness of the initialization strategy for neurOps. Recall that we first initialize all neurOps with standard operators and then finetune their parameters in a later joint training step. We compare our choice with two alternatives: random initialization from scratch, initialization with standard operators without allowing further finetuning (*i.e.*, in later steps, only strength predictors are trained while neurOps keep fixed). The performance is given in Table 2 (a) and (d). The results verify that initialization using standard operators is clearly superior to random initialization (PSNR: 24.32db vs 22.61db). Besides, using trainable neurOps is also shown to be a better choice than using fixed functional standard operators (PSNR: 23.78db), due to better model expressiveness.

Recall that we choose the three most commonly used standard operators to initialize our neurOps. We then test whether the order of standard operators matters. As shown in Table 2 (b) and (d), we test all 6 permutations of different ordering, and find that initializing the first, second and third neural operators with `black clipping`, `exposure` and `vibrance`, respectively, is the best choice.

We further conduct experiments to find out how many neurOps are suitable. We test four choices:  $K = 1, 2, 3$  and  $4$ . For  $K = 1$ , we use `black clipping` for initialization. For  $K = 2$ , we use `black clipping` and `exposure` for initialization. For  $K = 4$ , we use another standard operator `highlight recovery` to initialize the 4th neurOp. We could find that  $K = 3$  achieves the best performance, as shown in Table 2 (c-d). This is possibly due to that fewer neural operators (*i.e.*,  $K = 1$  or  $2$ ) lead to insufficient expressiveness while a larger number of neural operators (*i.e.*,  $K = 4$ ) result in harder training of strength predictors due to longer sequence. Overall,  $K = 3$  is the best choice.

**Strength Predictors.** In the global pooling layer of a strength predictor, recall that we have combined three pooling functions that compute average, maximum, and standard deviation, respectively. We test cases when a part of or all pooling

functions are used, as shown in Table 3 (a) and (d). Notice that each pooling function is useful, and combining all of them achieves the best performance.

We also verify other design choices of strength predictors in Table 3 (b). Recall that we feed intermediate images from previous editing steps as the input to strength predictors. An alternative choice would be always feeding the original input image, however, that would result in a drop in performance, *i.e.*, PSNR: 24.32db → 23.91db. It verifies that following human’s editing workflow to use intermediate visual feedbacks is a better choice. Besides, recall that we force all strength predictors to share parameters for convolutional layers. We also test the case when parameters are not forced sharing. Unsurprisingly, the performance slightly gains, *i.e.*, increasing PSNR by 0.03db. However, it doubles the network size ( $28k \rightarrow 56k$ ). Sharing parameters for all strength predictors leads to a good trade-off between maintaining high performance and lightweight structures.

**Loss Function.** Our loss function (Equation 5) includes a reconstruction loss term, a TV loss term, and a color loss term. In Table 3 (c), we evaluate the choice of terms in the loss function. The results verify that combining all terms produces the best performance.

## 6 Conclusion and Limitations

In this paper, we have proposed a lightweight sequential image retouching method. The core of our method is the newly introduced neural color operator. It mimics the behavior of traditional color operators and learns complex pixel-wise color transformation whose strength is controlled by a scalar. We have also designed CNN based strength predictors to automatically infer the scalar strengths. The neural color operators and strength predictors are trained in an end-to-end manner together with a carefully designed initialization scheme.

Extensive experiments show that our model achieve the state-of-the-art performance on public datasets with only 28k parameters and provide more convenient parametric controls compared with previous competitive works.

**Limitations.** Our method has several limitations and could be further improved in the future. First, our method is limited to global color retouching. A possible way to handle local effects is extending strength predictors to infer strength maps instead of scalar strengths. Second, we only support pixel-wise color editing. It would be an interesting topic to extend neural color operators to handle spatial filtering. Third, from a theoretical aspect, our encoder-decoder structure for neural color operators cannot guarantee accurate homomorphism properties. It is worthwhile to investigate other network structures with better theoretical properties, such as invertible networks. As for future works, we are also interested in applying the idea of neural color operators to other related applications such as image editing.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China (Project Number: 61932003).

## References

1. Afifi, M., Derpanis, K.G., Ommer, B., Brown, M.S.: Learning multi-scale photo exposure correction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
2. Aly, H.A., Dubois, E.: Image up-sampling using total-variation regularization with a new observation model. *IEEE Transactions on Image Processing* **14**(10), 1647–1659 (2005)
3. Arici, T., Dikbas, S., Altunbasak, Y.: A histogram modification framework and its application for image contrast enhancement. *IEEE Transactions on Image Processing* **18**(9), 1921–1935 (2009)
4. Aubry, M., Paris, S., Hasinoff, S.W., Kautz, J., Durand, F.: Fast local laplacian filters: Theory and applications. *ACM Transactions on Graphics (TOG)* **33**(5), 1–14 (2014)
5. Bianco, S., Cusano, C., Piccoli, F., Schettini, R.: Content-preserving tone adjustment for image enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 0–0 (2019)
6. 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 Conference on Computer Vision and Pattern Recognition (CVPR). pp. 97–104. IEEE (2011)
7. Chai, Y., Giryes, R., Wolf, L.: Supervised and unsupervised learning of parameterized color enhancement. In: The IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 992–1000 (2020)
8. Chen, H.T., Wei, L.Y., Chang, C.F.: Nonlinear revision control for images. *ACM Trans. Graph.* **30**(4) (jul 2011). <https://doi.org/10.1145/2010324.1965000>
9. 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 Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6306–6314 (2018)
10. Deng, Y., Loy, C.C., Tang, X.: Aesthetic-driven image enhancement by adversarial learning. In: 2018 ACM Multimedia Conference on Multimedia Conference (MM). pp. 870–878. ACM (2018)
11. 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), 118 (2017)
12. Grabler, F., Agrawala, M., Li, W., Dontcheva, M., Igarashi, T.: Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph.* **28**(3) (jul 2009). <https://doi.org/10.1145/1531326.1531372>, <https://doi.org/10.1145/1531326.1531372>
13. 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)
14. He, J., Liu, Y., Qiao, Y., Dong, C.: Conditional sequential modulation for efficient global image retouching. In: European Conference on Computer Vision (ECCV). pp. 679–695. Springer (2020)
15. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the AAAI conference on artificial intelligence (2018)

16. 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)
17. Hwang, S.J., Kapoor, A., Kang, S.B.: Context-based automatic local image enhancement. In: European Conference on Computer Vision (ECCV). pp. 569–582. Springer (2012)
18. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: DSLR-quality photos on mobile devices with deep convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 3277–3285 (2017)
19. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: WESPE: weakly supervised photo enhancer for digital cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 691–700 (2018)
20. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). pp. 1125–1134 (2017)
21. Jiang, Y., Gong, X., Liu, D., Cheng, Y., Fang, C., Shen, X., Yang, J., Zhou, P., Wang, Z.: EnlightenGAN: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing (TIP)* **30**, 2340–2349 (2021)
22. Jimenez Rezende, D., Eslami, S., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N.: Unsupervised learning of 3D structure from images. *Advances in neural information processing systems* **29**, 4996–5004 (2016)
23. 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)
24. Kim, H.U., Koh, Y.J., Kim, C.S.: PieNet: Personalized image enhancement network. In: European Conference on Computer Vision (ECCV). pp. 374–390. Springer (2020)
25. 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 (October 2021)
26. Kim, Y.T.: Contrast enhancement using brightness preserving bi-histogram equalization. *IEEE transactions on Consumer Electronics* **43**(1), 1–8 (1997)
27. Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. arXiv preprint arXiv:1503.03167 (2015)
28. Lee, C., Lee, C., Kim, C.S.: Contrast enhancement based on layered difference representation of 2d histograms. *IEEE transactions on image processing* **22**(12), 5372–5384 (2013)
29. Li, C., Guo, C., Ai, Q., Zhou, S., Loy, C.C.: Flexible piecewise curves estimation for photo enhancement (2020)
30. 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 (June 2021)
31. Liang, J., Zeng, H., Zhang, L.: High-resolution photorealistic image translation in real-time: A laplacian pyramid translation network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
32. Liu, E., Li, S., Liu, S.: Color enhancement using global parameters and local features learning. In: Asian Conference on Computer Vision (ACCV) (2020)
33. Liu, Y., He, J., Chen, X., Zhang, Z., Zhao, H., Dong, C., Qiao, Y.: Very lightweight photo retouching network with conditional sequential modulation. CoRR **abs/2104.06279** (2021), <https://arxiv.org/abs/2104.06279>

34. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3431–3440 (2015)
35. Moran, S., Marza, P., McDonagh, S., Parisot, S., Slabaugh, G.: DeepLPF: Deep local parametric filters for image enhancement. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12826–12835 (2020)
36. Moran, S., McDonagh, S., Slabaugh, G.: CURL: Neural curve layers for global image enhancement. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 9796–9803. IEEE (2021)
37. Ni, Z., Yang, W., Wang, S., Ma, L., Kwong, S.: Towards unsupervised deep image enhancement with generative adversarial network. *IEEE Transactions on Image Processing (TIP)* **29**, 9140–9151 (2020)
38. Park, J., Lee, J.Y., Yoo, D., So Kweon, I.: Distort-and-Recover: Color enhancement using deep reinforcement learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5928–5936 (2018)
39. Shi, J., Xu, N., Xu, Y., Bui, T., Dernoncourt, F., Xu, C.: Learning by planning: Language-guided global image editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13590–13599 (2021)
40. Stark, J.A.: Adaptive image contrast enhancement using generalizations of histogram equalization. *IEEE Transactions on image processing* **9**(5), 889–896 (2000)
41. Wang, B., Yu, Y., Xu, Y.Q.: Example-based image color and tone style enhancement. *ACM Transactions on Graphics (TOG)* **30**(4), 1–12 (2011)
42. 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 Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6849–6857 (2019)
43. 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 (October 2021)
44. Wang, Y., Chen, Q., Zhang, B.: Image enhancement based on equal area dualistic sub-image histogram equalization method. *IEEE transactions on Consumer Electronics* **45**(1), 68–75 (1999)
45. Yan, J., Lin, S., Bing Kang, S., Tang, X.: A learning-to-rank approach for image color enhancement. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2987–2994 (2014)
46. Yan, Z., Zhang, H., Wang, B., Paris, S., Yu, Y.: Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)* **35**(2), 11 (2016)
47. Ying, Z., Li, G., Ren, Y., Wang, R., Wang, W.: A new low-light image enhancement algorithm using camera response model. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (CVPRW). pp. 3015–3022 (2017)
48. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: Learning enriched features for real image restoration and enhancement. In: ECCV (2020)
49. 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)
50. Zhao, L., Lu, S.P., Chen, T., Yang, Z., Shamir, A.: Deep symmetric network for underexposed image enhancement with recurrent attentional learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12075–12084 (October 2021)