FlexiCurve: Flexible Piecewise Curves Estimation for Photo Retouching

Chongyi Li[1*] Chunle Guo[2*] Shangchen Zhou[1] Qiming Ai[1] Ruicheng Feng[1] Chen Change Loy[1†]
[1]S-Lab, Nanyang Technological University [2]VCIP, CS, Nankai University
{chongyi.li,ccloy}@ntu.edu.sg, guochunle@nankai.edu.cn,
{s200094,qai001,ruicheng002}@e.ntu.edu.sg
https://li-chongyi.github.io/FlexiCurve

(a) FlexiCurve-trained with **paired** data **(b)** FlexiCurve-trained with **unpaired** data

Figure 1. **Visual examples by FlexiCurve.** The first row represents the low-quality images sampled from Adobe 5K dataset [3]. The second row represents the corresponding results by the proposed FlexiCurve, where **(a)** represents the results retouched by FlexiCurve trained with paired data and **(b)** represents the results retouched by FlexiCurve trained with unpaired data. FlexiCurve can deal with global tone and local properties well and does not introduce over-/under-enhancement regions regardless of paired or unpaired training data.

Abstract

This paper presents a new method, called FlexiCurve, for photo retouching. Unlike most existing methods that perform image-to-image mapping, which requires expensive pixel-wise reconstruction, FlexiCurve takes an input image and estimates global curves to adjust the image. The adjustment curves are specially designed for performing piecewise mapping, taking nonlinear adjustment and differentiability into account. To cope with challenging and diverse properties in real-world photos, FlexiCurve is formulated to produce diverse estimations. The spatial dependencies among these estimations are implicitly modeled by a Transformer structure to improve local retouching of different regions. Thanks to the image-to-curve formulation, FlexiCurve only needs a lightweight network. Our method improves efficiency without compromising the retouching quality and losing details in the original image. The method is also appealing as it is not limited to paired training data, thus it can flexibly learn rich retouching styles from unpaired data. Extensive experiments demonstrate the efficiency, retouching performance, and flexibility of our method quantitatively and qualitatively.

1. Introduction

Photo retouching that improves the appearance (e.g., color, tone, brightness, saturation) of an image enjoys remarkable progress thanks to deep learning. Despite the impressive performance of current methods, they suffer from some limitations. 1) In general, large networks are required to cope with pixel-wise reconstruction, which in-

evitably leads to a high memory footprint and long inference time due to massive parameter space. 2) Improving the efficiency using shallow networks will compromise the ability to preserve or enhance local details. Shallow networks may even generate artifacts and artificial colors due to their limited capacity. **3)** Many existing methods [1,10,11,13,16,24,30,31,31,34,35,38] require paired training data, and thus they can only produce a fixed retouching style. Relying on paired training data increases the risk of overfitting specific data and limits the flexibility of these models to meet diverse user requirements. Clearly, trading-off efficiency, retouching performance, and flexibility is still an open research problem in photo retouching.

To address the aforementioned problems, we draw inspiration from existing photo editing tools. In these tools, users usually remap the tonality of an image by adjusting a curve that specifies a function from the input level to the output level of a specific color channel. For instance, one can adjust the highlight of the image by moving a point in the top portion of the curve or adjust the shadows by changing a point in the bottom section of the curve. Moving a point up and down lightens or darkens the tonal area while dragging a point left to right changes the contrast. Curve adjustment can be made more flexible by adding more control points (or knot points) to adjust different tonal areas.

We found such curve adjustment properties appealing for deep learning-based photo retouching [2, 11, 25]. Instead of generating an entire image as an output, a curve estimation network only needs to estimate a handful of curve parameters. Retouching can be achieved by just curve mapping, which simplifies the retouching network and makes the whole process more efficient than conventional pixel-wise reconstruction. Curve adjustment can preserve details and inherent relations of neighboring pixels well through the direct mapping of input level to output level.

Despite the attractive properties of curve adjustment, devising a deep curve estimation network for photo retouching is non-trivial. A naïve global curve is not desired since it treats the whole image homogeneously. Nonetheless, estimating a transformative coefficient for every single pixel is too expensive. The challenge is to devise a network and the corresponding output space to cater for piecewise curves, so as to balance the flexibility of curves and efficiency. The process is akin to placing control points in the right position on a curve and moving them in the correct direction and level. Another challenge is to cope with diverse properties and illuminations in different regions of an image. Piecewise curve adjustment is still global, and thus the results may experience over enhancement in the bright regions, under enhancement in the dark regions, or producing artifacts.

In this work, we formulate a new method called **FlexiCurve** that enjoys the benefits of curve adjustment yet is capable of overcoming the aforementioned challenges. FlexiCurve takes an image as input to estimate the corresponding adjustment curves. To allow flexibility in curve estimation, we design the output space to encompass both the knot points to constrain the adjustment range of each sub-function of a piecewise curve and the associated curve parameters to control the curvature.

To cope with diverse and heterogeneous regions in images, we further allow the network to produce multiple adjustment curves for curve mapping. This gives us multiple solutions. Modeling the spatial relationship of these solutions allows us to handle both global and local properties. The modeling of spatial dependencies is made possible by using a simplified Transformer structure.

Thanks to the succinct design, the proposed FlexiCurve only consists of 130K trainable parameters, which is orders of magnitude smaller than popular network structures such as U-Net [29] (31,042K trainable parameters) that is used for image enhancement [4, 7]. Its inference speed is about 0.024s on an NVIDIA 1080Ti GPU for an image of $512\times512\times3$. Some representative results of our method are shown in Figure 1, suggesting the capability of FlexiCurve in returning visually pleasing results with and without using paired data for training.

We summarize the contributions as follows. (1) We present an approach that performs photo retouching by piecewise curves estimation through a deep network. Unlike existing methods that spend expensive computation for pixel-wise reconstruction, the new notion of curve adjustment enables parameter-efficient network structure, fast inference, and good details preservation and enhancement. (2) We explain how piecewise adjustment curves can be designed as the network's output. We show that a flexible curve can be realized through knot points and nonlinear curve parameters, which collectively perform piecewise and nonlinear adjustment while being differentiable in the process of gradient back-propagation.

2. Related Work

Early works on photo retouching mainly perform color adjustment, ranging from example-based methods [15, 28] to retrieval-based methods [19]. These methods may suffer from potential visual artifacts due to improper exemplars.

Learning-based methods are popular for photo retouching. They spread from Convolutional Neural Network (CNN)-based, Generative Adversarial Network (GAN)-based, to Reinforcement Learning (RL)-based methods. To be specific, CNNs were employed to approximate image processing operators [5], reconstruct latent images [16, 39], estimate affine transforms [11, 32, 36], compute the nodes of the spline [2], learn parametric filters [24], estimate the neural curve [25], or estimate multi-layer perceptrons [13] by supervised learning on paired training data.

Unpaired adversarial learning based on the GAN frame-

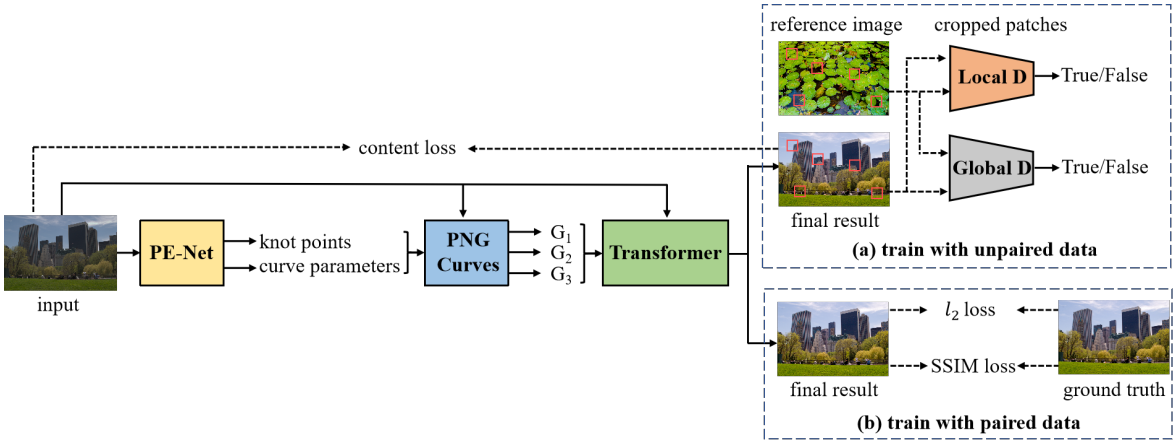


Figure 2. **Overview of the FlexiCurve framework.** The input image is first fed to the parameter estimation network (PE-Net) for estimating a set of knot points and curve parameters. The knot points and curve parameters define the piecewise nonlinear global curves (PNG Curves), which are used to adjust the level of the RGB channels of an input image. Multiple curves are generated simultaneously to provide us with multiple globally adjusted results (G_1, G_2, G_3). The final result is achieved by blending such intermediate results via a Transformer. **(a) FlexiCurve trained with unpaired data,** where global and local discriminators are used to distinguish whether the final image or randomly cropped patches are 'real' or 'fake', respectively. To preserve the original image content, a content loss is employed. **(b) FlexiCurve trained with paired data,** where the ℓ_2 and $SSIM$ losses are employed for supervised training.

work has been used for photo retouching. For example, a two-way GAN (i.e., DPE) was proposed to enhance images by pixel-wise reconstruction [7]. Deng et al. [8] proposed an EnhanceGAN for automatic image enhancement, which requires binary labels on image aesthetic quality. Ni et al. [26] proposed to learn an image-to-image mapping from a set of images with desired characteristics in an unsupervised manner for photo enhancement. In addition, Park et al. [27] proposed a reinforcement learning-based color enhancement method, Distort-and-Recover, to learn the optimal global enhancement sequence of retouching actions.

In comparison to existing deep learning-based photo retouching models, we propose to solve this problem differently using the following novel sequence of steps. 1) Instead of performing image-to-image mapping [5,7,8,13,16,23,26], we perform image-to-curves mapping then use the piecewise curves for adjustment. The whole process can be trained end-to-end as we design the curves to be differentiable in the process of gradient back-propagation. The formulation allows FlexiCurve to enjoy good retouching performance, a lightweight network structure, and fast inference speed. It not only improves the robustness in coping with diverse properties in local regions of images but also offers the flexibility for a user to explore different retouching options. 2) Different from parameter estimation-based methods [2,11,24,25], our method estimates optimal piecewise curve parameters, in which we nonlinearly adjust the curvature of each sub-function of a curve, thus improving the flexibility of curve mapping. Moreover, our method is not limited to paired training data, thus it can flexibly learn different retouching styles.

3. Methodology

We present the overview of our FlexiCurve framework with unpaired training data in Figure 2(a). The framework can be easily extended to supervised learning if paired data are available. The FlexiCurve framework with paired training data is presented in Figure 2(b). As illustrated, the proposed framework is specially designed to allow flexibility in curve estimation. This is achieved through a Parameter Estimation Network (PE-Net) that estimates a set of optimal knot points and curve parameters. The framework then separately maps the pixels in RGB channels of a given image by the Piecewise NOnlinear Global curves (PNG Curves) defined by knot points and curve parameters.

To deal with diverse image content properties and illumination of different regions in the input image, the framework produces a set of piecewise nonlinear global curves, each of which retouches regions in the image differently. Given multiple sets of global adjustment curves, our approach generates multiple globally adjusted results (each result consists of three adjusted RGB channels). The final result can be achieved by fusing these globally adjusted results spatially via a Transformer network that models their spatial dependencies.

3.1. Piecewise Nonlinear Global Curve

Drawing inspiration from previous works [12,20,21,25], we design a PNG Curve that allows **1)** adjustment of input pixels in a piecewise manner, which constrains the adjustment range of each sub-function of a piecewise curve by some knot points, **2)** nonlinear mapping in each piece, which controls the curvature by estimating associated curve

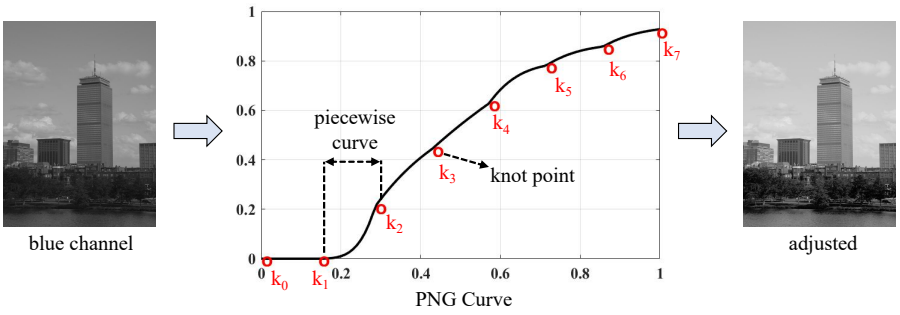


Figure 3. **An illustration of PNG Curve global adjustment.** The black line represents a PNG Curve, wherein the red points represent the estimated knot points and the curve between two knot points is a piecewise curve. The horizontal axis and the vertical axis represent the input and output pixel values, respectively.

parameters, **3)** end-to-end training as the curve is differentiable. The formulation of the PNG Curve can be expressed as:

$$V = k_0 + \sum_{m=0}^{M-1} (k_{m+1} - k_m)\mathbf{S}(I; m, \alpha_m), \tag{1}$$

$$\mathbf{S}(I; m, \alpha_m) = \mathbf{F}_c^n(\mathbf{F}_\delta(IM - m); \alpha_m), \tag{2}$$

where $I \in \mathbb{R}^{h\times w}$ in the range of [0,1] and $V \in \mathbb{R}^{h\times w}$ are the input channel and globally adjusted channel, respectively. The height and width of I and V are represented by h and w, respectively. The variable k_m denotes the value of mth knot point and $k_{m+1} - k_m$ constrains the adjustment ranges in the corresponding piece. The total number of pieces is represented by M, thus leading to $M + 1$ knot points. The curvature of nonlinear curve in each piece is controlled by $\alpha \in$[-1,1] and α_m is the nonlinear curve parameter of the mth piece. Both k and α are learned by the PE-Net. For the estimation of k, we first equally divide the horizontal axis and then estimate the value of each knot point, as shown in Figure 3.

The mth piecewise curve is represented by $(k_{m+1} - k_m)\mathbf{S}(I; m, \alpha_m)$ that can be derived as in Eq. (2), where $\mathbf{F}_c^n$ controls the nonlinearity in each piece, which can obtain a larger curvature by iteratively applying the basic curve $\mathbf{F_c}$ for n times. The basic curve $\mathbf{F}_c$ can be expressed as:

$$\mathbf{F}_c(x; \alpha) = x + \alpha x(1 - x), \tag{3}$$

where x is the input of $\mathbf{F}_c$. Thus, $\mathbf{F}_c^n$ can be expressed as:

$$\mathbf{F}_c^n = \mathbf{F}_c^{n-1} + \alpha\mathbf{F}_c^{n-1}(1 - \mathbf{F}_c^{n-1}), \tag{4}$$

where n is the number of iterations, which can further increase or decrease the curvature. More flexible curvature is important for challenging cases, such as extremely dark or over-exposed regions. We set the number of iterations n to 4 for the trade-off between performance and efficiency. We will investigate the effect of iteration numbers in the sensitivity analysis. The function $\mathbf{F}_\delta$ in Eq. (2) makes the PNG Curve successive and differentiable. It can be expressed as:

$$\mathbf{F}_\delta(y) = \begin{cases} 0 & y < 0 \\ y & 0 \le y \le 1, \\ 1 & y > 1 \end{cases} \tag{5}$$

where y denotes the input of the function $\mathbf{F}_\delta$.

We separately apply one PNG Curve to each of the three RGB channels of a given image and yield one globally adjusted result. We further allow the network to produce multiple adjustment curves for curve mapping. This gives us multiple complementary solutions and each performs differently. Curve mapping via multiple sets of PNG Curves produces multiple globally adjusted results, denoted as $G_1, G_2, G_3 \in \mathbb{R}^{h\times w\times c}$ in Figure 2. The number of channels c is set to 3 for RGB color image.

We typically have a small set of knot points and curve parameters for each PNG Curve, the estimation of which only requires a lightweight network. The effects of using the different number of knot points and curve parameters will be investigated in the sensitivity analysis. In Figure 3, we show an example of adjusting the blue channel of an image using a PNG Curve. As shown in Figure 3, the PNG Curve adjusts the input pixel values in a piecewise manner. It nonlinearly increases or decreases the dynamic range of input pixels in each piece. Improved brightness and contrast are observed after such global adjustment.

3.2. Parameter Estimation Network

The knot points and curve parameters are closely related for obtaining the final result. Consequently, we estimate them in a multi-task framework. We use a common backbone to extract shared features and fork different branches for the estimation of the aforementioned knot points and curve parameters. We design the PE-Net to be lightweight. One can replace the proposed PE-Net with a more powerful and complex network at the cost of higher computations. As a baseline model, we only adopt a plain and parameter-efficient network structure.

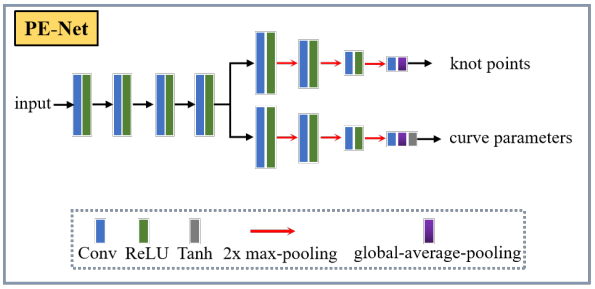


Figure 4. **The network structure of the proposed parameter estimation network (PE-Net).**

As shown in Figure 4, PE-Net only consists of 12 convolutional layers and each layer consists of 32 convolutional kernels of size 3×3 and stride 1. Except for the last layer in each branch that outputs knot points and curve parameters, respectively, all convolutional layers are followed by the ReLU activation function. Instead of the commonly used fully connected layers for parameter estimation in neural networks [22, 25], the knot points and curve parameters are obtained by global average pooling after the last convolutional layer, which reduces the computational resource costs and relaxes the sizes of the input image. We constrain the curve parameter α of each piece in the range of [-1,1] by using the Tanh activation function. In both the knot point estimation branch and the curve parameter estimation branch, we employ the 2× max-pooling operations to enlarge the receptive field and reduce the computational burden.

We adopt eight knot points and seven sets of curve parameters for each PNG Curve. There is a PNG Curve for each of the three RGB channels. We assume N globally adjusted results. Thus, the two branches of PE-Net produce $N\times3\times8$ knot points and $N\times3\times7$ sets of curve parameters.

3.3. Transformer

Curve adjustment performs global adjustment and thus falls short in considering different characteristics of local regions. To avoid over-/under-enhancement in local regions, we formulate our method to produce multiple globally adjusted results, called global solutions for brevity in this part. The different characteristics in different regions of multiple global solutions are modeled via a simplified Transformer structure [9] to achieve better global and local retouching.

Specifically, as shown in Figure 5, the input $\in \mathbb{R}^{h\times w\times 3}$ and three global solutions ($G_1, G_2, G_3 \in \mathbb{R}^{h\times w\times 3}$) are fed to four parallel convolutional layers followed by the ReLU activation function for transforming them into feature space. After that, we obtain $Q \in \mathbb{R}^{h\times w\times l}$ (Query), $K_1 \in \mathbb{R}^{h\times w\times d}$ (Key 1), $K_2 \in \mathbb{R}^{h\times w\times d}$ (Key 2), and $K_3 \in \mathbb{R}^{h\times w\times d}$ (Key 3) of the Transformer, respectively. The height and width of the features are represented by h and w, respectively. The number of feature channels l and d is set to 96 and 32, respectively. We also treat G_1, G_2, and G_3 as Value 1, Value 2, and Value 3 of the Transformer, respectively. To model their spatial dependencies, we separately employ a single-head attention module:

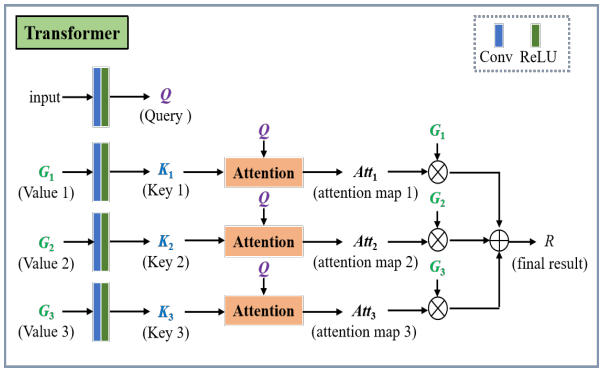


Figure 5. **The network structure of the proposed Transformer.** The global solutions G_1, G_2, and G_3 are treated as Value 1, Value 2, and Value 3, respectively. We separately transform the input and three global solutions into feature space and then obtain Query, Key 1, Key 2, and Key 3. The final result can be achieved by modeling the spatial dependencies among input and global solutions.

$$\widetilde{G}_i = \text{softmax}(QK_i^\top)G_i, \tag{6}$$

where $\widetilde{G} \in \mathbb{R}^{h\times w\times 3}$ represents the locally highlighted G. The variable i is 1, 2, or 3. The term softmax$(QK^\top)$ represents an attention map (denoted as $Att \in \mathbb{R}^{h\times w\times 3}$ in Figure 5). The final result $R \in \mathbb{R}^{h\times w\times 3}$ can be achieved by

$$R = \sum_{i=1}^{n} \widetilde{G}_i, \tag{7}$$

where n is set to 3 by default in our method.

3.4. Loss Functions

Training with Unpaired Data. We adopt adversarial training to train our model with unpaired data. We have two discriminators i) a vanilla global discriminator D_g that determines whether the image is from the target manifold, i.e., a reference image collection and ii) a local discriminator D_l that learns to distinguish between random local patches from the generator's output and arbitrarily sampled images from the target manifold. We iteratively update our network and discriminators in the training phase. The total loss of generator (i.e., our forward network) can be expressed as:

$$L_{unpair}^{total} = W_g L_{G_g} + W_l L_{G_l} + L_c, \tag{8}$$

where W_g and W_l are the weights of the global adversarial loss L_{G_g} and local adversarial loss L_{G_l}, respectively. The content loss (perceptual loss [17]) is represented as L_c.

Training with Paired Data. When paired data is available for training, we use ℓ_2 loss and $SSIM$ loss [33] to train our model. The total loss can be expressed as:

$$L_{pair}^{total} = \ell_2 + W_{ssim} L_{ssim}, \tag{9}$$

Table 1. **Sensitivity analysis that refers to the network hyper-parameters (PE-Net), curve parameters (PNG Curve), number of global solutions (GS).** The final FlexiCurve is equipped with 32 output feature maps in each convolutional layer denoted as $f32$, 12 convolutional layers denoted as $l12$, 7 knot points in each PNG Curve denoted as $k7$, 4 iterations of the nonlinear adjustment curve in each piece denoted as $i4$, 3 global solutions denoted as $N3$, i.e., FlexiCurve-$f32$-$l12$-$k7$-$i4$-$N3$.

Components	Baselines	PSNR↑	SSIM↑
	FlexiCurve-pair ($f32$-$l12$-$k7$-$i4$-$N3$)	25.34	0.93
PE-Net	$f8$-$l12$	23.33	0.88
	$f16$-$l12$	23.92	0.88
	$f64$-$l12$	25.51	0.93
	$f32$-$l24$	25.68	0.93
PNG Curve	$k7$-$i1$	24.02	0.89
	$k7$-$i6$	24.91	0.92
	$k10$-$i4$	25.55	0.93
	$k16$-$i4$	24.65	0.91
	$k32$-$i4$	24.25	0.91
GS	$N1$ (global)	23.37	0.88
	$N2$	25.15	0.92
	$N4$	25.21	0.92
	$N5$	25.10	0.92

where ℓ_2 and L_{ssim} are ℓ_2 and SSIM losses, respectively, the weight of $SSIM$ loss is represented as W_{ssim}.

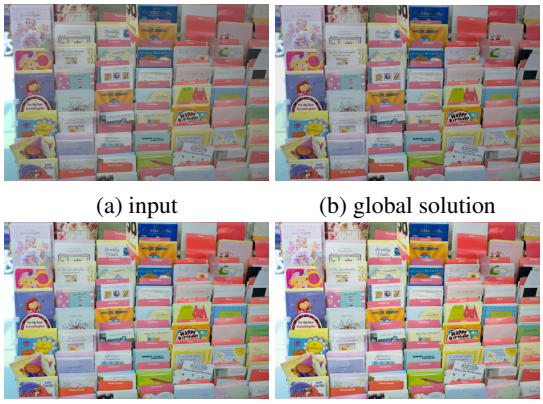
4. Experiments

4.1. Experimental Settings

Training and Testing Sets. We conduct experiments on two commonly used benchmark datasets for photo retouching. All images used in our study are 8bit sRGB images.

First, we use the same train and test sets as Chen et al. [7]. We use the MIT-Adobe 5K dataset [3] that contains 5,000 images and the corresponding reference images retouched by five experts. Following Chen et al. [7], we split the dataset into three partitions: 2,250 images and their retouched versions are used for paired training, the retouched versions of another 2,250 images are treated as the reference results for unpaired training, and the rest 500 images are used for testing. We denote the 500 pairs of testing data as MIT-Adobe5k-DPE in this paper. We randomly select 500 images from the paired training set as the validation set. The results retouched by expert C are treated as reference images. All images in the MIT-Adobe 5K dataset are decoded into the png format and resized to have a long-edge of 512 pixels using Lightroom as the processing in Chen et al. [7]. Our method can process any size of images.

Second, we follow Park et al. [27] and Zhao et al. [39]'s processing to output the data from the MIT-Adobe 5K dataset [3]. We use the same training and testing data as Zhao et al. [39], in which the first 4,500 images are used for training while the last 500 images are used for testing. The testing dataset is denoted as MIT-Adobe5k-ICCV21.



(a) input (b) global solution (c) final result (d) ground truth

Figure 6. **A visual comparison between global solution and our final result.**

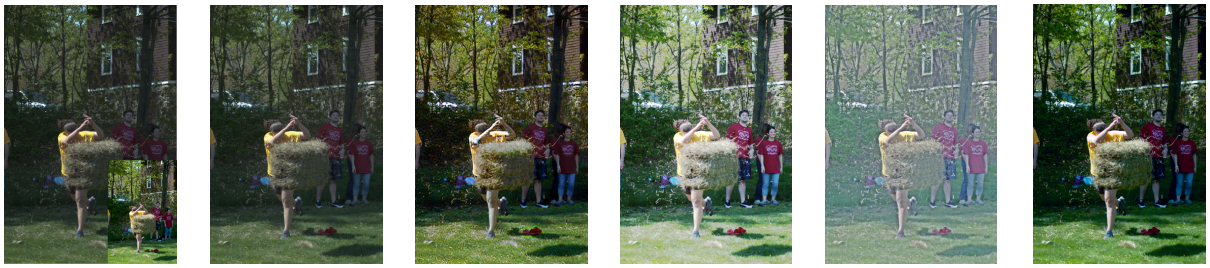
Implementation Details. We implement our framework with PyTorch on an NVIDIA Tesla V100 GPU. We set the batch size to 4 and randomly crop the images into a size of 256×256 during training. We use ADAM [18] optimizer with default parameters for optimizing our frameworks. The global and local discriminators adopt the same network structure as the discriminator used in CartoonGAN [6] based on its stability and efficiency. We use the fixed learning rate $5e^{-5}$ for our generator and discriminators optimization. Both weights W_g and W_l are set to 6 to balance the scale of losses. For FlexiCurve with paired data (denoted as **FlexiCurve-pair**), we set the initial learning rate to $1e^{-4}$ and then decreased it to $1e^{-5}$ after 100 epochs until convergence. We set the weight W_{ssim} to 1.

Baselines. We compare the proposed method with related works/tool, including PhotoShop Auto, FCN [5], CycleGAN [40], DPE [7], White-Box [14], Zero-DCE [12], CSRNet [13], DeepLPF [24], and Zhao et al. [39]. We retrain the compared methods (excluding the reinforcement learning-based White-Box [14]) if their training data are different from ours. We found the splits of training and testing data used in DeepLPF [24][1] are different from the original splits of MIT-Adobe5k-DPE dataset used in DPE [7] while the original splits are commonly used for training and testing in previous works. We also retrain our method using the same splits as DeepLPF [24]. Our retrained model is denoted as FlexiCurve-pair*. For the method of PhotoShop Auto, we select Adjustment → Curves → Options → Enhance Brightness and Contrast in Photoshop software.

4.2. Sensitivity Analysis

We use the FlexiCurve-pair framework to conduct sensitivity analysis based on its stable training. Quantitative comparisons are carried out on the validation set. We first

[1] https://github.com/sjmoran/DeepLPF/tree/master/adobe5k_dpe



(a) input/Expert C (b) PhotoShop Auto (c) CycleGAN [40] (d) White-Box [14] (e) Zero-DCE [12] (f) FlexiCurve-unpair

Figure 7. **Visual comparisons of the unsupervised models on MIT-Adobe5K-DPE dataset.**

quantify the effects of network hyperparameters, curve parameters, and number of global solutions in Table 1.

Effect of Network Hyperparameters. We first study the effects of network hyperparameter settings in our PE-Net, consisting of the network's depth and width. In the baselines of PE-Net, f represents the number of output feature maps in each convolutional layer (except for the last layer in each branch) and l represents the number of convolutional layers, where the feature extractor and two branches have the same numbers of convolutional layers (i.e., $l/3$).

As presented in Table 1, increasing more feature maps f from 8 to 32 (our final choice) improves the quantitative performance while increasing from 32 to 64 only slightly improves the performance at the cost of more computational resources. When increasing the number of convolutional layers l from 12 to 24, we did not find obvious gains. Thus, we adopt $f32$ and $l12$ for PE-Net.

Effect of Curve Parameters. We investigate the effects of curve parameter settings in the PNG Curve, including the number of knot points (also related to the number of pieces) and iterations employed in the nonlinear adjustment curve in each piece. In the baselines of the PNG Curve, k represents the number of knot points in each PNG Curve and i represents the number of iterations of the nonlinear adjustment curve in each piece.

Observing Table 1, increasing the number of iterations from 1 to 4 significantly improves the values of PSNR and SSIM, suggesting the importance of flexible adjustment of curvature. However, further increasing the number of iterations from 4 to 6 cannot achieve obvious improvements because 4 iterations can handle the most challenging cases. Increasing the number of knot points, on the contrary, decreases the quantitative performance. The potential reason is that estimating more knot points increases the pressure of PE-Net under the limited network hyperparameters. Considering the minor differences between these ablated models, we adopt the settings of $k7$ and $i4$ for PNG Curve to balance computational resources and retouching performance.

Effect of the Number of Global Solutions. To demonstrate the effect of the number of global solutions, we provide the results of removing the Transformer, i.e., only using the global piecewise curve to retouch input image, denoted as $N1$. As shown in Table 1, single global retouching shows lower values of PSNR and SSIM when compared with the outputs produced by a Transformer.

(a) input (b) -Adobe5K (c) -HDR

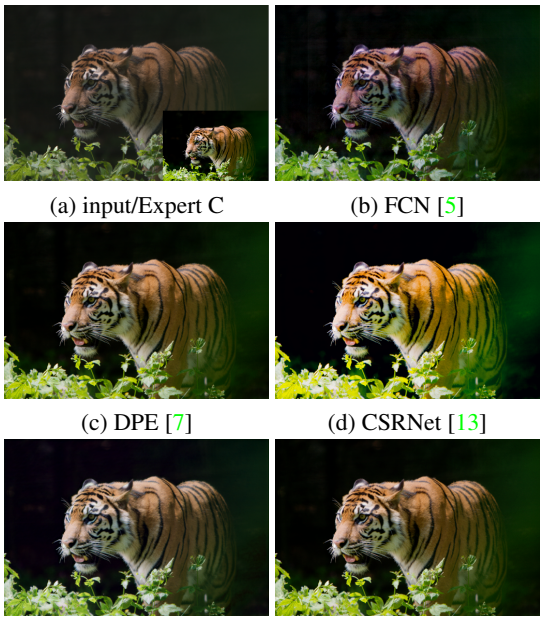
Figure 8. **Results of our unpaired models trained on (b) MIT-Adobe5K dataset (-Adobe5K) and (c) HDR dataset (-HDR).**

Although the global solution can improve the brightness and color of the input image, it is still a global adjustment, which leads to the loss of local details in the result shown in Figure 6(b). In contrast, the final result of FlexiCurve that fuses multiple global solutions has clear details, high contrast, and vivid color as shown in Figure 6(c). The visual and quantitative results validate our motivations of fusing multiple global solutions instead of only using a single global piecewise curve. However, we found that increasing the number of global solutions from 3 to 5 slightly decreases the quantitative performance. This is because producing more global solutions in our framework requires more piecewise curves and more complex structures, which is beyond the current network capacity.

4.3. Perceptual Comparisons

The visual comparisons on typical low-quality images sampled from on MIT-Adobe5K-DPE dataset are shown in Figure 7. As shown, our FlexiCurve-unpair effectively improves the visual quality of the input images. The results of our method have improved brightness, vivid color, and good visibility. In comparison to other unsupervised methods that introduce color deviations, over-exposure, and artifacts, our method produces more visually pleasing results.

To verify the flexibility of our method, we collect another unpaired training dataset to train the FlexiCurve with unpaired data. The new unpaired training dataset includes



(a) input/Expert C (b) FCN [5]

(c) DPE [7] (d) CSRNet [13]

(e) Zhao et al. [39] (f) FlexiCurve-pair

Figure 9. **Visual comparisons of the supervised models trained on the same splits on the MIT-Adobe5K-DPE dataset.**

one low-quality image dataset sampled from the low-quality image of the MIT-Adobe5K [3] dataset and one High Dynamic Range (HDR) dataset collected from Flickr. A set of visual results are shown in Figure 8. Compared with the results retouched by the FlexiCurve-unpair trained on the MIT-Adobe5K dataset (unpaired input/Expert C data), the results of the FlexiCurve-unpair trained on the new unpaired dataset (unpaired input/HDR data) enjoy higher contrast and more vivid color that are in line with the characteristics of HDR data. We provide a video demo of our FlexiCurve-unpair trained on this new dataset for retouching a video. The video can be found on YouTube at https://www.youtube.com/watch?v=BFWP0KSA7JI.

We further show the results of different supervised models in Figure 9, where all compared methods are trained on the same training dataset. FCN [5] and CSRNet [13] change the color of the foreground. Zhao et al. [39] cannot recover the brightness and color. Our FlexiCurve-pair achieves good contrast and clear details without color deviations and over-enhancement.

4.4. Quantitative Comparisons

We carry out quantitative comparisons using PSNR, SSIM [33], and LPIPS [37] and also report the trainable parameters of different deep models. In Tables 2 and 3, our FlexiCurve-pair achieves the best average values of PSNR, SSIM, and LPIPS across different testing datasets, which indicates the good content, structure, and perceptual similarity between our results and the ground truth images. The performance of our FlexiCurve-pair is superior to the pixel-wise reconstruction-based method DPE [7], DeepLPF [24], and Zhao et al. [39] that come with a large network. Apart from the supervised model, our FlexiCurve-unpair also obtains good quantitative performance as presented in Table 2. Although CSRNet [13] has the smallest network trainable parameters, its quantitative performance is inferior to some compared methods. Our method achieves the third smallest network trainable parameters, having only 130K.

Table 2. **Quantitative comparisons in terms of PSNR (in dB), SSIM, and LPIPS on MIT-Adobe5k-DPE dataset, and the trainable parameters (#P, in K).** '-' indicate the result is not available. The top rows present the results of the models trained using unpaired data or without training data. The middle rows show the results of the models trained using paired data and the same splits of training and testing data as DPE [7]. The bottom rows show the results of the models trained using the same splits as DeepLPF [24]. FlexiCurve-pair* indicates the retrained model of our method on the same splits as DeepLPF [24].

Method	PSNR↑	SSIM↑	LPIPS↓	#P↓
input	17.82	0.78	0.15	-
PhotoShop Auto	20.16	0.80	0.13	-
CycleGAN [40]	20.73	0.78	0.27	11,380
White-Box [14]	19.21	0.82	0.14	8,560
Zero-DCE [12]	12.44	0.68	0.24	79
FlexiCurve-unpair	**21.98**	**0.87**	**0.09**	130
FCN [5]	23.10	0.88	0.08	750
DPE [7]	23.80	0.90	0.07	3,350
CSRNet [13]	21.73	0.88	0.08	**36**
Zhao et al. [39]	23.34	0.89	0.07	11,560
FlexiCurve-pair	**24.03**	**0.91**	**0.06**	130
DeepLPF [24]	23.93	0.90	0.06	1,800
FlexiCurve-pair*	**24.37**	**0.92**	**0.06**	130

Table 3. **Quantitative comparisons in terms of PSNR (in dB), SSIM, and LPIPS on MIT-Adobe5k-ICCV21 dataset.**

Method	PSNR↑	SSIM↑	LPIPS↓
input	17.70	0.79	0.15
FCN [5]	23.61	0.88	0.07
CSRNet [13]	24.36	0.90	0.06
DeepLPF [24]	23.63	0.88	0.07
Zhao et al. [39]	24.27	0.90	0.06
FlexiCurve-pair	**24.74**	**0.92**	**0.06**

5. Conclusion

We proposed a curve estimation method for photo retouching. Photos are retouched via specially designed piecewise curves coupled with spatial dependencies. The estimation of curve parameters is achieved within a multi-task network. The spatial dependencies among multiple estimations are modeled by a Transformer. Experiments show the effectiveness and efficiency of our method.

Acknowledgement. This study is supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

References

[1] Mahmoud Afifi, Konstantinos G. Derpanis, Bjorn Ommer, and Michael S. Brown. Learning multi-scale photo exposure correction. In *CVPR*, pages 9157–9167, 2021. 2

[2] Simone Bianco, Claudio Cusano, Flavio Piccoli, and Raimondo Schettini. Personalized image enhancement using neural spline color transforms. *TIP*, 29:6223–6236, 2020. 2, 3

[3] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Fredo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR*, pages 97–10, 2011. 1, 6, 8

[4] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *CVPR*, pages 3291–3300, 2018. 2

[5] Qifeng Chen, Jia Xu, and Vladlen Koltun. Fast image processing with fully-convolutional networks. In *CVPR*, pages 2497–2506, 2017. 2, 3, 6, 8

[6] Yang Chen, Yukun Lai, and Yongjin Liu. CartoonGAN: Generative adversarial network for photo cartoonization. In *CVPR*, pages 9465–9474, 2018. 6

[7] Yusheng Chen, Yuching Wang, Manhsin Kao, and Yungyu Chuang. Deep photo enhancer: Unpaired learning for image enhancement form photographs wigh gans. In *CVPR*, pages 6306–6314, 2018. 2, 3, 6, 8

[8] Yubin Deng, Chen Change Loy, and Xiaoou Tang. Aesthetic-driven image enhancement by adversarial learning. In *ACM MM*, pages 870–878., 2018. 3

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5

[10] Qiqi Gao, Jie Li, Tiejun Zhao, and Yadong Wang. Real-time image enhancement with attention aggregation. *ACM MCCA*, 19(25). 2

[11] Michael Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Fredo Durand. Deep bilateral learning for real-time image enhancement. *ACM TOG*, 36(4):1–12, 2017. 2, 3

[12] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runming Cong. Zero-reference deep curve estimation for low-light image enhancement. In *CVPR*, pages 1780–1789, 2020. 3, 6, 7, 8

[13] Jingwen He, Yihao Liu, Yu Qiao, and Chao Dong. Conditional sequential modulation for efficient global image retouching. In *ECCV*, pages 679–695, 2020. 2, 3, 6, 8

[14] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM TOG*, 37(2):1–17, 2018. 6, 7, 8

[15] Youngbae Hwang, Joon Young Lee, In So Kweon, and Seon Joo Kim. Color transfer using probabilistic moving least squares. In *CVPR*, pages 3342–3349, 2014. 2

[16] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. DSLR-quality photos on mobile devices with deep convolutional networks. In *ICCV*, pages 3277–3285, 2017. 2, 3

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016. 5

[18] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Joon Young Lee, Kalyan Sunkavalli, Zhe Lin, Xiaohui Shen, and In So Kweon. Automatic content-aware color and tone stylization. In *CVPR*, pages 2470–2478, 2016. 2

[20] Chongyi Li, Ruicheng Feng Chunle Guo, Shangchen Zhou, and Chen Change Loy. CuDi: Curve distillation for efficient and controllable exposure adjustment. *arXiv preprint arXiv:2207.14273*, 2022. 3

[21] Chongyi Li, Chunle Guo, and Chen Change Loy. Learning to enhance low-light image via zero-reference deep curve estimation. *TPAMI*, 2021. 3

[22] Jiahao Li, Bin Li, Jizheng Xu, Ruiqing Xiong, and Wen Gao. Fully connected network-based intra prediction for image coding. *TIP*, 27(7):3236–3247, 2018. 5

[23] Zhexin Liang, Chongyi Li, Shangchen Zhou, Ruicheng Feng, and Chen Change Loy. Iterative prompt learning for unsupervised backlit image enhancement. *arXiv preprint arXiv:2303.17569*, 2023. 3

[24] Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. DeepLPF: Deep local parametric filters for image enhancement. In *CVPR*, pages 12826–12835, 2020. 2, 3, 6, 8

[25] Sean Moran, Gregory Slabaugh, Steven McDonagh, and Gregory Slabaugh. CURL: Neural curve layers for global image enhancement. In *ICPR*, pages 9796–9803, 2020. 2, 3, 5

[26] Zhangkai Ni, Wenhan Yang, Shiqi Wang, Lin Ma, and Sam Kwong. Towards unsupervised deep image enhancement with generative adversarial network. *TIP*, 29:9140–9151, 2020. 3

[27] Jongchan Park, Joon Yound Lee, Donggeun Yoo, and In So Kweon. Distort-and-Recover: Color enhancement using deep reinforcement learning. In *CVPR*, pages 5928–5936, 2018. 3, 6

[28] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001. 2

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 2

[30] Yuda Song, Hui Qian, and Xin Du. StarEnhancer: Learning real-time and style-aware image enhancement. In *ICCV*, pages 4126–4135, 2021. 2

[31] Ardhendu Shekhar Tripathi, Martin Danelljan, Samarth Shukla, Radu Timofte, and Luc Van Gool. Transform your smartphone into a DSLR camera: Learning the isp in the wild. pages 1–17, 2022, booktitle =. 2

[32] Tao Wang, Yong Li andJingyang Peng, Yipeng Ma, Xian Wang, Fenglong Song, and Youliang Yan. Real-time image enhancer via learnable spatial-aware 3D lookup tables. In *ICCV*, pages 2471–2480, 2021. 2

[33] Zhou Wang, Alan C Bovik, and Hamid R Sheikh Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 5, 8

[34] Runsheng Xu, Zhengzhong Tu, Yuanqi Du, Xiaoyu Dong, Jinlong Li, Zibo Meng, Jiaqi Ma, Alan Bovik, and Hongkai Yu. Pik-fix: Restoring and colorizing old photos. In *WACV*, 2023. 2

[35] Canqian Yang, Meiguang Jin, Yi Xu, Rui Zhang, Ying Chen, and Huaida Liu. SepLUT: Separable image-adaptive lookup tables for real-time image enhancement. pages 1–17, 2022, booktitle =. 2

[36] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3D lookup tables for high performance photo enhancement in real time. *TPAMI*, 2020. 2

[37] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 8

[38] Zhaoyang Zhang, Yitong Jiang, Jun Jiang, Xiaogang Wang, Ping Luo, and Jinwei Gu. STAR: A structure-aware lightweight transformer for real-time image enhancement. In *ICCV*, pages 4106–4115, 2021. 2

[39] Lin Zhao, Shaoping Lu, Tao Chen, Zhenglu Yang, and Ariel Shamir. Deep symmetric network for underexposed image enhancement with recurrent attentional learning. In *ICCV*, pages 12075–12084, 2021. 2, 6, 8

[40] Junyan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 6, 7, 8