StarEnhancer: Learning Real-Time and Style-Aware Image Enhancement

Yuda Song [1] Hui Qian [1] Xin Du [2*]
[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China
{syd,qianhui,duxin}@zju.edu.cn

Abstract

Image enhancement is a subjective process whose targets vary with user preferences. In this paper, we propose a deep learning-based image enhancement method covering multiple tonal styles using only a single model dubbed StarEnhancer. It can transform an image from one tonal style to another, even if that style is unseen. With a simple one-time setting, users can customize the model to make the enhanced images more in line with their aesthetics. To make the method more practical, we propose a well-designed enhancer that can process a 4K-resolution image over 200 FPS but surpasses the contemporaneous single style image enhancement methods in terms of PSNR, SSIM, and LPIPS. Finally, our proposed enhancement method has good interactability, which allows the user to fine-tune the enhanced image using intuitive options.

1. Introduction

The development of smartphone cameras has dramatically lowered the barriers to take photos, but amateurs still lack the skills to get high-quality photos. To this end, a variety of image post-processing techniques have been proposed to bridge the gap. Generally, these techniques tend to improve the quality of image detail, which is broadly objective. However, the quality of a photo depends not only on the image detail but also on whether the photo meets people's aesthetics, which is entirely subjective. Therefore, various deep learning-based approaches [7, 15, 22] are proposed to retouch photos to make these photos more aesthetically pleasing. But only a few works [12, 25] have realized the difference between image enhancement and other low-level vision tasks. In our opinion, a practical image enhancement method should not employ a generic image restoration network but aim to be real-time and style-aware.

It is straightforward that different users have different aesthetic preferences, so the target image's tonal style is not constant [19, 25, 28]. And different cameras have different camera response functions (CRFs) and image signal processing (ISP) pipeline [1, 14], which means that the tonal style of the input image is also not constant. Meanwhile, users may want to transform the retouched images into their preferred style, whereas the unretouched images are not available. Therefore, we consider that a practical image enhancement method requires the capacity to transform images between multiple tonal styles. Finally, because user preferences are challenging to quantify precisely, the intuitive manual adjustment options are appealing [16, 29].

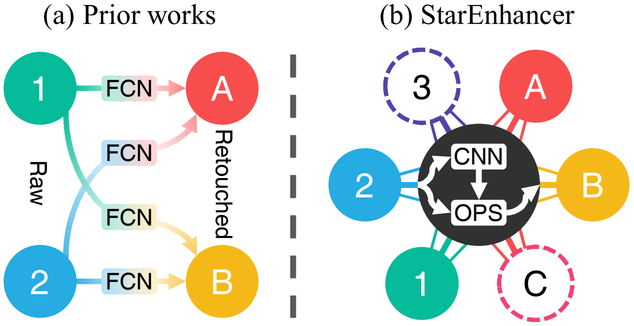


Figure 1: Comparison between prior works and our proposed method. (a) Prior works need to train multiple models to learn the mappings between source styles and target styles. (b) StarEnhancer offers the capability to transform images from one style to another using a single model.

The current common image restoration strategy is to train a fully convolutional network (FCN) [33] to reconstruct the images fed into the network. However, the computational complexity of the FCN grows quadratically with the spatial dimensions of the input images [18]. And the FCN-based network is more difficult to train and may introduce artifacts [12], especially when employing generative adversarial networks (GANs) [3, 7, 13]. Besides, the FCN's receptive field does not change with the input image's size, which may lead to a visible dissimilarity between the enhanced images of the same image in different resolutions [12]. In contrast, color transform-based image en-

[*] Corresponding author.

hancement methods [12, 49] only use convolutional neural networks (CNNs) to encode the color transformation's parameters from a fixed-size, low-resolution version of the image. And the learned color transformation functions can be applied to the full-resolution images, whose computational complexity is extremely low. Therefore, we consider the color transform-based image enhancement methods may be plausible solutions.

As illustrated in Figure 1 (a), most existing image enhancement methods need to train an individual FCN for each transformation. More critically, a new dataset needs to be collected for each new style transformation, yet collecting a dataset for image enhancement is challenging since it relies on expert knowledge [4, 46]. Also, considering the high computational complexity of FCN when processing high-resolution images, such methods can only provide limited capabilities to users.

To this end, we propose a more practical image enhancement method that is real-time and style-aware to bridge these gaps. We named our method StarEnhancer to admire StarGAN [8, 9], albeit our approach is vastly different from StarGAN. As demonstrated in Figure 1 (b), StarEnhancer utilizes multiple tonal styles' training data and learns the mapping between multiple tonal styles using a single model. Specifically, we first train a style classifier to classify images and take the output embedding vector of the classifier's penultimate layer as the latent codes. The mapping network then encodes these latent codes as a set of style codes, which customizes the curve encoder using Dual AdaIN modified from adaptive instance normalization (AdaIN) [20]. The curve encoder predicts the curves' parameters from the low-resolution version of the image, and the enhancer applies these curves to transform the full-resolution image. Unlike the existing color transform-based image enhancement methods [5, 12, 24, 28, 31, 38, 46, 49], StarEnhancer considers the correlation between color channels and the pixel's coordinates.

Overall, our contributions are as follows:

- We propose a highly efficient curve-based enhancer that can enhance a 4K-resolution image over 200 FPS on a single GPU. And it is scale-invariant and artifact-free, which is critical for high-resolution images.
- We propose a flexible approach named StarEnhancer for image enhancement between multiple styles. It can be customized to meet different camera characteristics and user preferences via a simple one-time setup.
- StarEnhancer provides intuitive options to allow users to fine-tune the results for each image manually.
- StarEnhancer achieves state-of-the-art performance in terms of efficiency and effectiveness on the MIT-Adobe-5K dataset [4].

2. Basic enhancer

The effectiveness and efficiency of the image enhancer can substantially affect the approach's practicality. Therefore, we first discuss how to design an expressive and fast image enhancer. Finally, we propose the basic enhancer for the single style transformation.

2.1. Problem formulation

Unlike the real-time image classification methods that emphasize network architecture design [18, 42, 52], the real-time image restoration methods also involve the impact of the input image's size on the processing efficiency. Deep learning-based image restoration methods are generally based on the FCNs, making their computational complexity quadratic to the input image's spatial dimensions. And most image enhancement methods also employ FCN-like network architectures [2, 6, 7, 16, 27, 48]. If using the network $\mathbf{G}$ with parameters θ to enhance the input image I directly, the output image O can be formulated as follows:

$$O = \mathbf{G}(I; \theta). \tag{1}$$

Fortunately, the standard image enhancement task is more aware of the global information, making it possible to utilize the down-sampled image to obtain informative features, just like the strategy applied in the high-level vision task. As a trade-off, it isn't easy to apply this kind of method to the detail-concerned image restoration tasks, including even the low-light image enhancement task [30, 34, 43, 47, 53]. Specifically, the network $\mathbf{G}$ with parameters θ extracts the features from the down-sampled input image $I{\downarrow}$, and these features are used to formulate the transformation function $\mathbf{F}$ applied to the input image I as follows:

$$O = \mathbf{F}(I; \mathbf{G}(I{\downarrow}; \theta)). \tag{2}$$

In this way, the backbone network's computational complexity for extracting features hardly varies with the input image size. And the key to designing a powerful image enhancer is to develop an efficient and expressive transformation function.

2.2. Prior art

There are roughly three categories of functions to follow: color transformation matrix [5, 12, 31], curve-based color transformation function [15, 24, 29, 38], and 3D lookup table (LUT) [49].

The color transformation matrix is a 3×4 affine transformation matrix that maps the pixel's input color to the output color. As an example, HDRNet [12] predicts low-resolution affine color transformation coefficient matrices in the bilateral grid. Guided by the full-resolution single-channel guide map, these matrices are sliced into full-resolution coefficient matrices that are then applied to the original image.

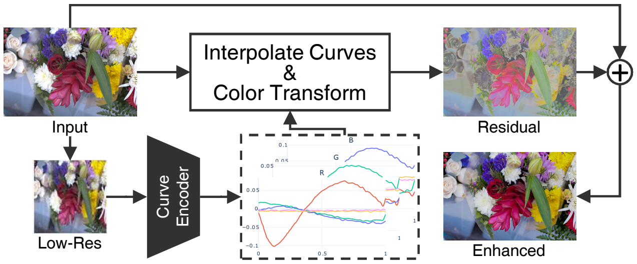


Figure 2: Framework of the proposed basic image enhancer.

However, the color transformation matrix's capability is not sufficient. And the guide map is generated by an FCN built with several point-wise convolution layers, which is still expensive for high-resolution images.

Curve-based color transformation functions mimic the color adjustment curve tool in retouching software (*e.g.* Lightroom and Photoshop) and are more in line with human retouching. In order to quantify a curve using a limited number of parameters, the backbone networks regress the knot points of the curve [24, 29, 38] or the coefficients of a pre-defined function (*e.g.* gamma function, polynomial function) [15, 19, 39]. However, existing curve-based color transformation functions mostly neglect the relationship between color channels, which is in line with the color adjustment curve tool's characteristics but lacks the capability to approximate complex transformations.

3D LUTs are expressive operators that have been used in ISP [23]. Generally, 3D LUTs are obtained by expert adjustment and are fixed after the adjustment. To this end, Adaptive 3DLUT [49] learns a set of basis 3D LUTs from the training dataset and uses a CNN to predict content-dependent weights from the down-sampled image. These weights are used to fuse multiple basis 3D LUTs into a single 3D LUT that is then applied to the full-resolution image's transformation. Sincerely, 3D LUT is an attractive transformation function. However, Adaptive 3DLUT is a trade-off approach because only the weights for fusing the basis 3D LUTs are adaptive to the input image, while the basis 3D LUTs are still fixed after training. We believe that it is not flexible enough to be used as the basic image enhancer for multi-style image enhancement.

2.3. Multi-curve enhancer

We consider building our enhancer based on a curve-based color transformation function. For the prior curve-based image enhancement methods, the color transformation for output channel $\mathrm{j} \in \{\mathrm{r}, \mathrm{g}, \mathrm{b}\}$ can be formulated as:

$$O_{\mathrm{j}} = \mathbf{F}(I_{\mathrm{j}}; \mathbf{G}(I\downarrow; \theta)_{\mathrm{j}}). \tag{3}$$

Combined with the ideas of color transformation matrix and 3D LUT, we propose a revised curve-based transformation function to build our basic enhancer. We note that both the color transformation matrix and the 3D LUT take into account the correlation between the color channels. In other words, each channel of the output image is correlated with each channel of the input image. Besides, we believe that introducing the coordinate maps $\{\mathrm{x}, \mathrm{y}\}$ can make the transformation function more expressive. Thus the revised transformation for input pixel $I(x, y)$ $(I, x, y \in [0, 1])$ can be formulated as:

$$O_{\mathrm{j}}(x, y) = \mathbf{F}(I(x, y), x, y; \mathbf{G}(I\downarrow; \theta)_{\mathrm{j}}). \tag{4}$$

Figure 2 illustrates how our proposed image enhancer processes a high-resolution image with $H \times W$ resolution. Firstly, the CNN-based curve encoder predicts a parameter vector $\mathbf{u}$ for all curves' knot points from the down-sampled input image with $K \times K$ resolution. Then we split $\mathbf{u} = \mathbf{G}(\mathbf{I}\downarrow; \theta)$ into 15 subvectors, in which $\mathbf{u}_{\mathrm{i,j}}$ corresponds to the curve that maps input channel $\mathrm{i} \in \{\mathrm{r}, \mathrm{g}, \mathrm{b}, \mathrm{x}, \mathrm{y}\}$ to output channel $\mathrm{j} \in \{\mathrm{r}, \mathrm{g}, \mathrm{b}\}$. We propose an extremely fast curve-based transformation using piecewise cubic interpolation [11] and indexing to utilize the knot points' parameter vectors. Using the piecewise cubic interpolation function $\mathcal{S}^{M,N}$, we interpolate an M-dimensional vector $\mathbf{u}_{\mathrm{i,j}} = [u_{\mathrm{i,j},0}, ..., u_{\mathrm{i,j},M-1}]^T$ to an N-dimensional vector $\mathbf{v}_{\mathrm{i,j}} = [v_{\mathrm{i,j},0}, ..., v_{\mathrm{i,j},N-1}]^T$ as follows:

$$\mathbf{v}_{\mathrm{i,j}} = \mathcal{S}^{M,N}(\mathbf{u}_{\mathrm{i,j}}). \tag{5}$$

Let $\mathbf{v}_{\mathrm{i,j}}(k)$ be the k-th element $v_{\mathrm{i,j},k}$ of $\mathbf{v}_{\mathrm{i,j}}$, we apply following transformation to obtain the residual image R:

$$\begin{aligned} R_{\mathrm{j}}(x, y) = {} & \mathcal{S}^{M_{\mathrm{r,j}},2^D}(\mathbf{u}_{\mathrm{r,j}})(\lfloor I_{\mathrm{r}}(x, y) \cdot (2^D - 1) \rfloor) \\ & + \mathcal{S}^{M_{\mathrm{g,j}},2^D}(\mathbf{u}_{\mathrm{g,j}})(\lfloor I_{\mathrm{g}}(x, y) \cdot (2^D - 1) \rfloor) \\ & + \mathcal{S}^{M_{\mathrm{b,j}},2^D}(\mathbf{u}_{\mathrm{b,j}})(\lfloor I_{\mathrm{b}}(x, y) \cdot (2^D - 1) \rfloor) \\ & + \mathcal{S}^{M_{\mathrm{y,j}},H}(\mathbf{u}_{\mathrm{y,j}})(\lfloor y \cdot (H - 1) \rfloor) \\ & + \mathcal{S}^{M_{\mathrm{x,j}},W}(\mathbf{u}_{\mathrm{x,j}})(\lfloor x \cdot (W - 1) \rfloor), \end{aligned} \tag{6}$$

where $\lfloor \cdot \rfloor$ is the floor function and D denotes the color depth of each channel. In practice, we only require the interpolated vector $\{\mathbf{v}_{\mathrm{i,j}}\}_{\mathrm{i} \in \{\mathrm{x}, \mathrm{y}\}}$ to be expanded to a map with the same resolution as the input image since the coordinates of the pixels are monotonic. Besides, it is feasible to apply a low color depth transformation to a high color depth image to reduce the cost of indexing (*e.g.* $D = 8$ for 48-bit color image). In this way, we need to render the residual image instead of rendering the enhanced image for preserving the information of high color depth images. Finally, the enhanced image can be obtained by $O = R + I$.

Given the image pair $\{I_a, I_b\}$, where I_a is the input image and I_b is the reference image, we compute the L_1 loss in CIELab color space to train the enhancer:

$$\mathcal{L}_E = \| Lab(I_b) - Lab(\mathbf{F}(I_a; \mathbf{G}(I_a\downarrow; \theta))) \|_1. \tag{7}$$

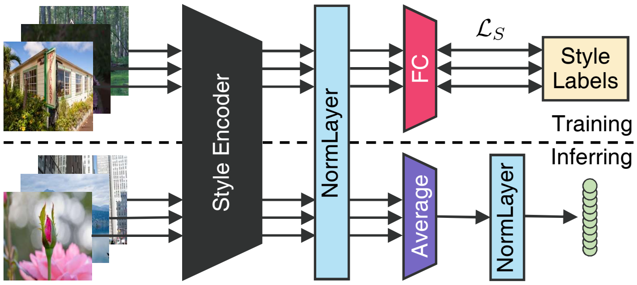


Figure 3: Framework of our proposed style encoder. When inferring, the input images are all from the same style.

3. StarEnhancer

In this section, we illustrate how to guide the enhancer to perform adaptive multi-style color transformations. There are two critical issues to be addressed: how to make the method adaptive to unseen styles and how to feed style information into the network.

3.1. Style encoder

We first discuss how to design the style code to encode unseen styles, thereby making the enhancer adaptive to new cameras and users. Apparently, a fixed label such as the one-hot vector used in StarGAN [8] is not a good choice. In contrast, the latent code used in StarGAN v2 [9] is a better choice, but it is obtained by randomly sampling from a known distribution, which only ensures diversity but does not establish a clear link to the style. Inspired by face recognition works [10, 32, 44, 45], we strive to train a style encoder that can learn image embeddings to establish the link between the specific style and the latent code.

Figure 3 illustrates an overview of our proposed approach. Specifically, we first train an image classifier on a dataset containing images of multiple tonal styles. Given the embedding $\mathbf{f}$ of the downsampled input image after the final global pooling layer and the corresponding style class label p, the loss can be formulated as follows:

$$\mathcal{L}_S = -\log\left(\frac{\exp\left(\frac{\mathbf{f}^T\mathbf{w}_p}{\|\mathbf{f}\|_2\|\mathbf{w}_p\|_2}\cdot s\right)}{\sum_{q\in Q}\exp\left(\frac{\mathbf{f}^T\mathbf{w}_q}{\|\mathbf{f}\|_2\|\mathbf{w}_q\|_2}\cdot s\right)}\right), \tag{8}$$

where s is a scaling term, Q denotes the style classes set, $\mathbf{w}$ is the weight of the last fully connected layer without bias term, and $\|\cdot\|_2$ is the L^2-norm.

In the inference phase, we feed n images of the specific style into the style encoder and obtain the embeddings $\{\mathbf{f}_i\}_{i=1,\ldots,n}$ after the global pooling layer. We approximate the embedding of the specific style by calculating the average of the L^2-normalized embeddings:

$$\mathbf{f}_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{f}_i}{\|\mathbf{f}_i\|_2}. \tag{9}$$

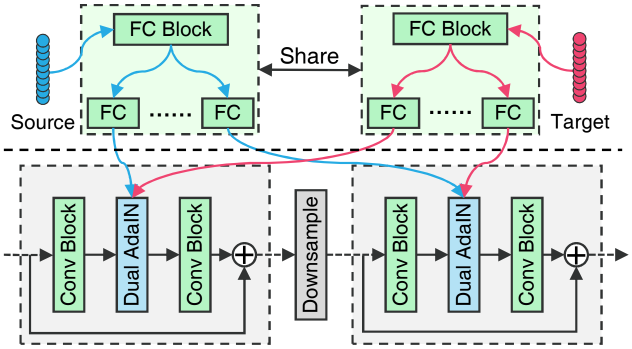


Figure 4: Framework of feeding the latent codes into the curve encoder. The top is the mapping network, while the bottom is the curve encoder with Dual AdaINs.

Since the average embedding does not always fall on the unit sphere as the single L^2-normalized embedding, we also apply L^2-normalization to it as follows:

$$\tilde{\mathbf{f}} = \frac{\mathbf{f}_{\text{avg}}}{\|\mathbf{f}_{\text{avg}}\|_2}. \tag{10}$$

We treat $\tilde{\mathbf{f}}$ as the center embedding of the specific style, as well as the latent code of the style.

3.2. Multi-style enhancer

Since the style-specific latent code has been obtained, now we need to feed the latent code into the curve encoder. We suppose that image enhancement can be viewed as a special type of style transfer so that AdaIN [20] may be a choice worth employing. However, we find that the normalization layer in the convolution block always leads to poor performance. To this end, we propose Dual AdaIN, which does not calculate the mean and variance of the feature maps for each input sample, but the mean and variance are obtained by mapping the latent codes to the style codes.

Figure 4 illustrates how to extend the basic enhancer to StarEnhancer. Firstly, we inserted the Dual AdaIN into the curve encoder of the basic enhancer in Figure 2. Then we fetch the latent codes $\{\tilde{\mathbf{f}}_d\}_{d\in\{a,b\}}$ of the source style class a and the target style class b using the style encoder. Given the latent codes $\{\tilde{\mathbf{f}}_d\}_{d\in\{a,b\}}$, the mapping network maps them to L sets of style codes $\{\mu_{d,1},\sigma_{d,1},\ldots,\mu_{d,L},\sigma_{d,L}\}_{d\in\{a,b\}}$, which are then fed into the curve encoder via Dual AdaIN:

$$\mathcal{F}'_j = \sigma_{b,j}\left(\frac{\mathcal{F}_j - \mu_{a,j}}{\sigma_{a,j}}\right) + \mu_{b,j}, \tag{11}$$

where $j \in \{1,\ldots,L\}$, $\mathcal{F}$ is the input feature map, and $\mathcal{F}'_j$ is the transformed feature map.

We also compute the $\mathcal{L}_E$ loss to train the multi-style enhancer, except that the training pair $\{I_a, I_b, \tilde{\mathbf{f}}_a, \tilde{\mathbf{f}}_b\}$ is randomly sampled from all possible styles ($a, b \in Q$).

3.3. User awareness

If the curve encoder and mapping network are trained using only the center embeddings of specific styles in the train set, they may tend to overfit these embeddings. To this end, we use subsets of the train set to generate more style embeddings, *i.e.*, feed fewer images of the specific style to generate additional style embeddings via Eq.(9) and Eq.(10).

New users can select their preferred images in the shared gallery or use their collection to generate new target latent codes. And the latent code of the source style can be pre-generated by the camera manufacturer or obtained using several unretouched images. Note that paired images are not necessary for this procedure.

We further provide manual fine-tuning options, which is very useful when the results do not meet user preference. For experts, all knot points of the predicted curves can be adjusted manually, just like the curve tool in Lightroom. But such a curve tool is still too difficult for non-experts, so we further propose a slider-based manual fine-tuning tool. Specifically, the user can adjust the sliders that correspond to $\{\beta_{\mathrm{i,j}}\}$ for tuning the contribution of each curve:

$$\mathbf{u}'_{\mathrm{i,j}} = \beta_{\mathrm{i,j}} \cdot \mathbf{u}_{\mathrm{i,j}}. \tag{12}$$

We use $\{\mathbf{u}'_{\mathrm{i,j}}\}$ to generate new curves and apply them to transform the image. Because our proposed curve-based enhancer is highly efficient and the manual fine-tuning procedure does not perform CNN inference, users can get feedback in real-time and further adjust the sliders.

3.4. Implementation detail

We build our StarEnhancer using PyTorch [40], and all operations used in the enhancer have been efficiently and differentiably implemented. Both the style encoder and the curve encoder are built on shallow ResNet [17], but with all batch normalization layers [21] removed from the convolution blocks. For a stable training after removing the batch normalization layers, we apply the Fixup initialization [50] as well as a few architecture modifications to the network.

We first train the style encoder using $\mathcal{L}_S$ loss and obtain the latent code $\tilde{\mathbf{f}}_q$ for each style class q, and then train the mapping network and curve encoder using $\mathcal{L}_E$ loss. When training the style encoder, we set the scaling term s in $\mathcal{L}_S$ loss to a large constant and assign a much larger learning rate to the last fully connected layer. All models are trained using the Adam optimizer [26] with the cosine annealing strategy [35] but not warm restarts.

When inferring, the style encoder and mapping network are only performed only once initially, then the fetched style codes are stored for future use. Further, the users can upload the preferred images to the server, which returns the corresponding style codes, so that the user devices only need to keep the model weights for the curve encoder.

4. Experiments

4.1. Experimental setup

We train and evaluate our method on the MIT-Adobe-5K dataset [4], which is the only dataset that consists of images in multiple expert retouching styles. MIT-Adobe-5K contains 5000 images captured by DSLR, each corresponding to a total of 12 styles, including 5 expert retouching styles (Artist A/B/C/D/E), 4 camera input styles, and 3 auto-retouching styles. StarEnhancer is the first method that exploits all the data in the MIT-Adobe-5K to the best of our knowledge.

Single style enhancement: we follow the experimental setup of the MIT-Adobe-5K-UPE benchmark [46] to evaluate our method's performance. Specifically, we use the images of the default input style as input, the images retouched by Artist C as the ground truth, and split the dataset into 4500 training image pairs and 500 test image pairs. All images in the test set retain their original resolution, varying from 2160 × 1440 to 6048 × 4032. We evaluate our method quantitatively using PSNR, SSIM, and LPIPS [51] to compare with contemporaneous methods.

Multi-style enhancement: we extend the experimental setup of the MIT-Adobe-5K-UPE benchmark to 10 styles, including 5 expert retouching styles (A/B/C/D/E), 3 camera input styles (O/P/Q), and 2 auto-retouching styles (X/Y). Note that the style Y is not provided by MIT-Adobe-5K but generated using the latest version of Lightroom. Further, the 3 remaining styles provided by MIT-Adobe-5K, together with the other 5 newly generated auto-retouching styles, are used to test our methods' performance when applied to unseen styles. All images of these styles are also split into train sets and test sets, but these train sets are not actually involved in training.

4.2. Single style enhancement

We compare our method with contemporaneous methods on MIT-Adobe-5K-UPE as shown in Table 1. StarEnhancer outperforms all the compared methods in terms of PSNR, SSIM, and LPIPS while capable of multi-style enhancement. Moreover, our proposed enhancer can achieve even better performance if we train the basic enhancer without Dual AdaIN on the unexpanded MIT-Adobe-5K-UPE dataset. StarEnhancer introduces correlation between channels, which makes it more expressive than another curve-based enhancer named CURL [38]. Adaptive 3DLUT [49] achieves excellent performance because of the expressive 3DLUT, but its encoder only predicts fusion weights, which limits its further improvement. But Adaptive 3DLUT is still the fastest method because it consists of only interpolation and slicing operations with a very lightweight backbone. Fortunately, StarEnhancer is comparably efficient, and the gap between them is hard to perceive by users. Note that all

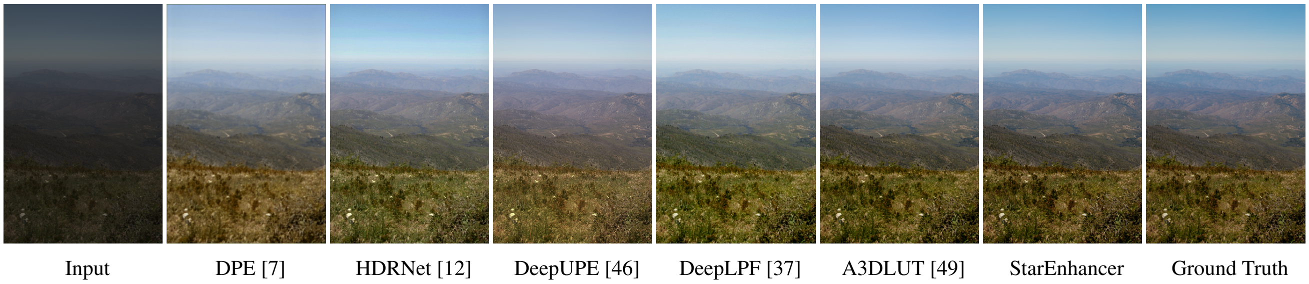


Figure 5: Qualitative comparison of single style transformation with contemporaneous methods on MIT-Adobe-5K-UPE.

Table 1: Quantitative comparison of single style transformation with contemporaneous methods on MIT-Adobe-5K-UPE. Running speed (FPS) is measured on 4K-resolution images using a single TITAN RTX. Note that some results are replicated from [37, 38, 49].

Method	PSNR	SSIM	LPIPS	FPS
Exposure [19]	18.57	0.701	–	0.11
Dis-Rec [39]	20.97	0.841	–	0.009
U-Net [41]	22.24	0.850	–	–
DPE [7]	22.15	0.850	–	
CURL [38]	24.20	0.880	0.108	
DeepLPF [37]	24.48	0.887	0.103	
HDRNet [12]	23.20	0.917	0.120	22
DeepUPE [46]	23.24	0.893	0.158	4.7
A3DLUT [49]	24.92	0.934	0.093	**602**
Basic	**25.46**	**0.948**	**0.083**	205
StarEnhancer	25.29	0.943	0.086	

U-Net-based methods are unable to enhance 4K-resolution images on a single GPU, which makes them impractical. Figure 5 further shows the qualitative comparison results for a single sample. It can be seen that the image enhanced by StarEnhancer is most similar to the ground truth in both tone and illumination, especially in the sky and grassland.

4.3. Multi-style enhancement

We first observe the distribution of features in the embedding space as shown in Figure 6. The expert retouching style and the camera input style are distributed on opposite ends of the sphere. Notably, the auto-retouching style X using the old Lightroom is close to the camera input style, while the auto-retouching style Y using the latest Lightroom is close to the expert retouching style, which can indicate the evolution of auto retouch tools. Further, Recall@1 illustrates that expert retouching styles are significantly more difficult to distinguish than camera input styles and auto-retouching styles, demonstrating that human aesthetics are subjective and difficult to quantify.

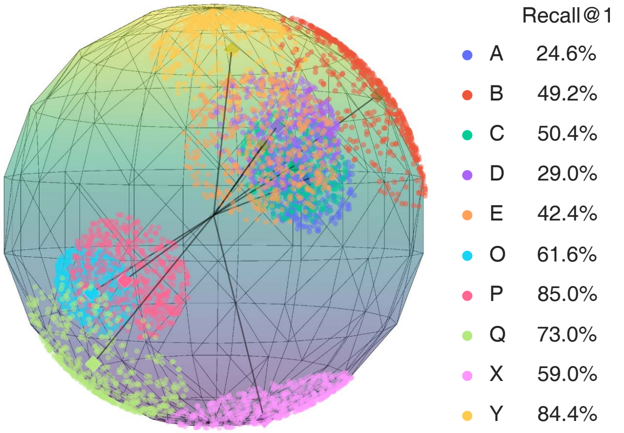


Figure 6: Visualization of features learned by style encoder, all features are projected onto the unit sphere using t-SNE [36]. Recall@1 for each class on the test set is listed.

We then focus on evaluating StarEnhancer's multi-style enhancement performance. Figure 7 shows some results of the mapping between multiple styles for a single sample. It is seen that StarEnhancer can adapt to different source styles, even if they vary greatly in brightness and color. Meanwhile, StarEnhancer can capture the characteristics of the target style to enhance the image. Specifically, the little girl in learned C's retouched image has more vivid clothes and a more natural-looking face. Finally, we find that the input style still affects the output image, such as the output images transformed from style P always have a cooler tone. This may be due to the insufficient variety of scenes in the MIT-Adobe-5K, which prevents the trained StarEnhancer from separating the style information well.

Figure 8 shows the quantitative evaluation results on the multi-style MIT-Adobe-5K benchmark. We do not add an explicit regular term to the loss function to constrain the same style's transformations, but StarEnhancer still performs impressively. The transformations between camera input styles are the simplest since only simple global color adjustments (*e.g.* white balance) are applied. In contrast,

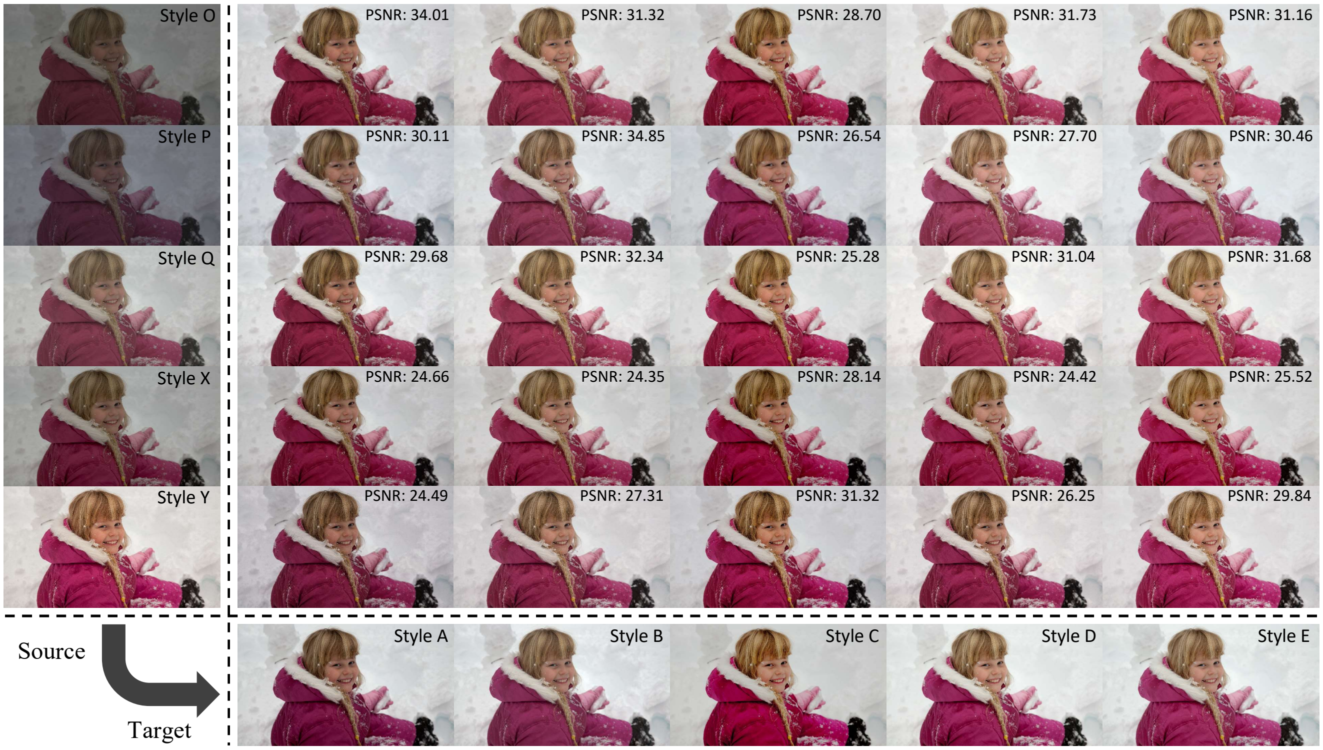
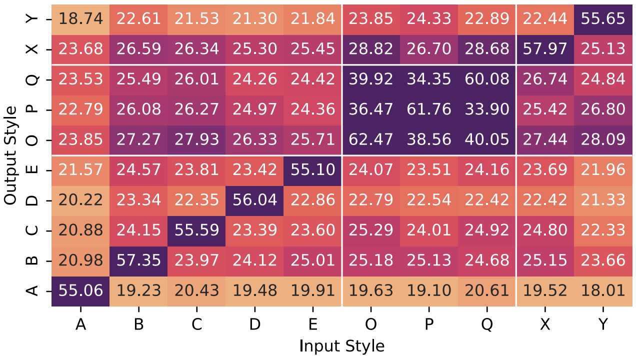


Figure 7: An example of mappings between multiple styles. The expert retouching styles are used as the target styles, and the other styles are used as the source styles.



	A	B	C	D	E	O	P	Q	X	Y
Y	18.74	22.61	21.53	21.30	21.84	23.85	24.33	22.89	22.44	55.65
X	23.68	26.59	26.34	25.30	25.45	28.82	26.70	28.68	57.97	25.13
Q	23.53	25.49	26.01	24.26	24.42	39.92	34.35	60.08	26.74	24.84
P	22.79	26.08	26.27	24.97	24.36	36.47	61.76	33.90	25.42	26.80
O	23.85	27.27	27.93	26.33	25.71	62.47	38.56	40.05	27.44	28.09
E	21.57	24.57	23.81	23.42	55.10	24.07	23.51	24.16	23.69	21.96
D	20.22	23.34	22.35	56.04	22.86	22.79	22.54	22.42	22.42	21.33
C	20.88	24.15	55.59	23.39	23.60	25.29	24.01	24.92	24.80	22.33
B	20.98	57.35	23.97	24.12	25.01	25.18	25.13	24.68	25.15	23.66
A	55.06	19.23	20.43	19.48	19.91	19.63	19.10	20.61	19.52	18.01

Figure 8: Quantitative results of the mappings between multiple styles of the MIT-Adobe-5K dataset [4] in PSNR.

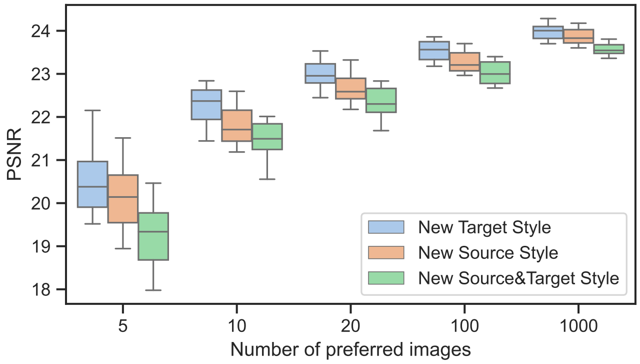


Figure 9: Quantitative results when StarEnhancer is applied to unseen styles.

transformations between expert retouching styles are much more difficult, and the most difficult style of all is expert retouching style A. Notably, we believe that the difficulty of learning styles is mainly related to the complexity of their transformations, while the recall of the style encoder indicates mostly the robustness of the transformation. Specifically, style Y is easier to distinguish but more difficult to learn than style X. We suppose this is because the new Lightroom's auto retouch tool is more complex and more robust.

4.4. Functional flexibility

The style encoder is introduced to enable StarEnhancer to perform enhancements between various unseen styles. We simulate selecting the users' preferred styles to evaluate starEnhancer's generalization performance using 8 unseen styles. We randomly choose a source style and a target style and select several image samples to generate new latent codes. Then we use the mapping network to transform these latent codes into style codes for Dual AdaIN. Finally,

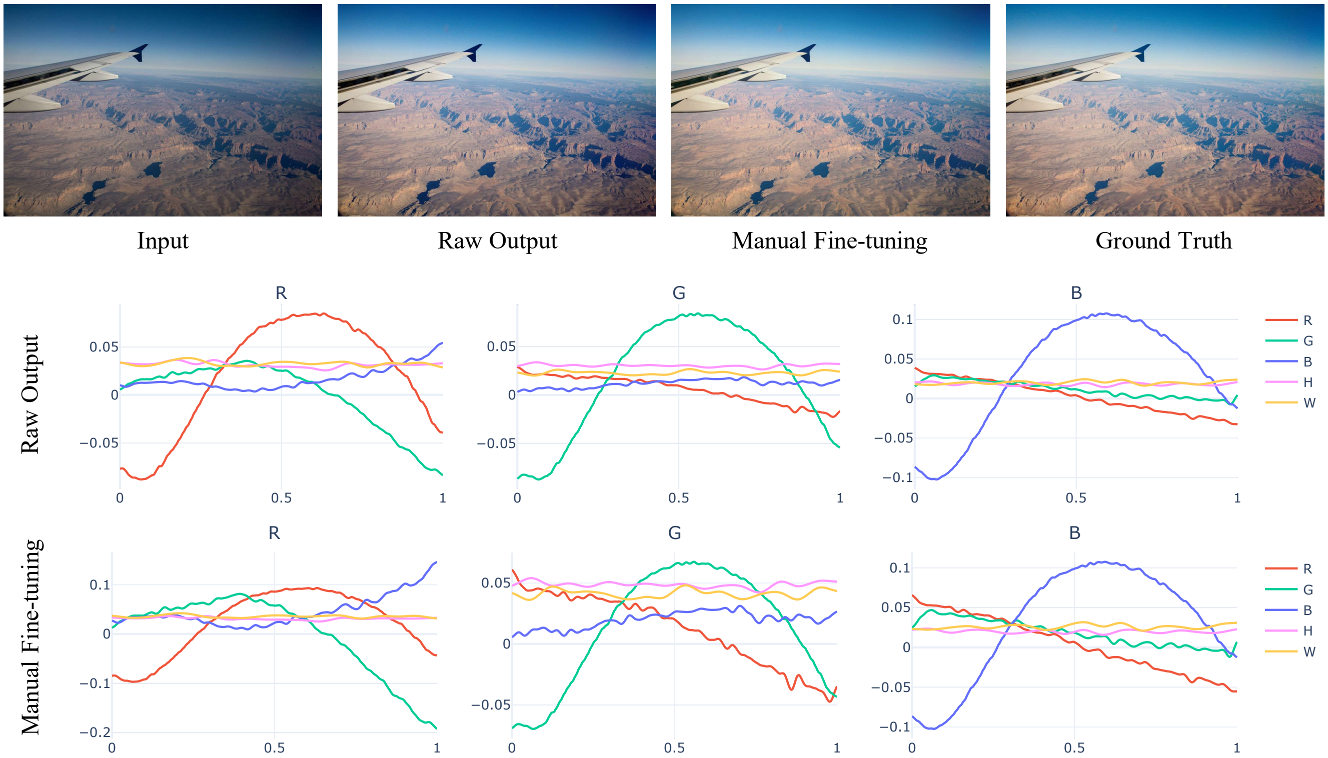


Figure 10: An example of manual fine-tuning, where users can obtain a more preferred result with the fine-tuning options. The top row lists the input image, the enhanced image output directly from StarEnhancer, the enhanced image after manual fine-tuning, and the ground truth image in the target style. The middle row shows the curves predicted by StarEnhancer. The bottom row shows the curves adjusted using the slider-based manual fine-tuning tool.

we evaluate the performance of the customized enhancer on the test sets. Because the number of samples affects the latent codes' reliability, we evaluate the enhancer's generalization performance in different sample size settings and repeat the experiment 10 times at each setting to reduce the error due to the difference in sample selection. As shown in Figure 9, the more samples used tend to yield more reliable latent codes, leading to better generalization performance. And generalizing to an unseen source style is more challenging than generalizing to an unseen target style. Furthermore, when both the source style and target style are unseen styles, the enhancer's performance decreases further. But even so, our enhancer can outperform most of the enhancers fine-tuned on the train sets of these styles.

Figure 10 shows an example of manual fine-tuning, and each curve's contribution can be observed. For this sample, the curves that map between the color channels contribute visibly to the residual image. The curves that map from the pixel's coordinates contribute little to the residual image. However, we believe that the pixel's coordinates are crucial, especially for some samples that are retouched using the gradient filter or elliptical filter. Because the enhanced image output directly from StarEnhancer differs significantly from the desired image, we manually adjust each curve's contribution using the proposed fine-tuning tool. Although the fine-tuning tool only stretches the curves, the fine-tuned image is significantly closer to the desired image.

5. Conclusion

In this paper, we propose an expressive curve-based image enhancer that can enhance a 4K-resolution image over 200 FPS. It surpasses the contemporaneous methods on the MIT-Adobe-5K. Based on our proposed style encoder and Dual AdaIN, we extend the enhancer to a multi-style enhancer and name it StarEnhancer, which can perform the mapping between multiple styles using a single model. Notably, our proposed approach is flexible enough to be applied to unseen styles. Lastly, we introduce a manual fine-tuning tool to meet the user preferences further.

Acknowledgement. This work is supported by National Key Research and Development Program of China under Grant 2020AAA0107400, Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ18F020002, Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies.

References

[1] Mahmoud Afifi and Michael S Brown. Deep white-balance editing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1397–1406, 2020.

[2] Mahmoud Afifi, Konstantinos G Derpanis, Bjorn Ommer, and Michael S Brown. Learning multi-scale photo exposure correction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9157–9167, 2021.

[3] Marc Bickel, Samuel Dubuis, and Sébastien Gachoud. Multiple generative adversarial networks analysis for predicting photographers' retouching. *arXiv preprint arXiv:2006.02921*, 2020.

[4] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 97–104. IEEE, 2011.

[5] Yoav Chai, Raja Giryes, and Lior Wolf. Supervised and unsupervised learning of parameterized color enhancement. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 992–1000, 2020.

[6] Qifeng Chen, Jia Xu, and Vladlen Koltun. Fast image processing with fully-convolutional networks. In *International Conference on Computer Vision (ICCV)*, pages 2497–2506, 2017.

[7] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6306–6314, 2018.

[8] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8789–8797, 2018.

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8188–8197, 2020.

[10] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2019.

[11] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[12] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 27:2672–2680, 2014.

[14] Michael D Grossberg and Shree K Nayar. Determining the camera response from images: What is knowable? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(11):1455–1467, 2003.

[15] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1780–1789, 2020.

[16] Jingwen He, Yihao Liu, Yu Qiao, and Chao Dong. Conditional sequential modulation for efficient global image retouching. In *European Conference on Computer Vision (ECCV)*, pages 679–695. Springer, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *International Conference on Computer Vision (ICCV)*, pages 1501–1510, 2017.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[22] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. Enlightengan: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing (TIP)*, 30:2340–2349, 2021.

[23] Hakki Can Karaimer and Michael S Brown. A software platform for manipulating the camera imaging pipeline. In *European Conference on Computer Vision (ECCV)*, pages 429–444. Springer, 2016.

[24] Han-Ul Kim, Young Jun Koh, and Chang-Su Kim. Global and local enhancement networks for paired and unpaired image enhancement. In *European Conference on Computer Vision (ECCV)*, pages 339–354. Springer, 2020.

[25] Han-Ul Kim, Young Jun Koh, and Chang-Su Kim. Pienet: Personalized image enhancement network. In *European Conference on Computer Vision (ECCV)*, pages 374–390. Springer, 2020.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[27] Yuma Kinoshita and Hitoshi Kiya. Convolutional neural networks considering local and global features for image enhancement. In *IEEE International Conference on Image Processing (ICIP)*, pages 2110–2114. IEEE, 2019.

[28] Dario Kneubuehler, Shuhang Gu, Luc Van Gool, and Radu Timofte. Flexible example-based image enhancement with

task adaptive global feature self-guided network. In *European Conference on Computer Vision (ECCV)*, pages 343–358. Springer, 2020.

[29] Chongyi Li, Chunle Guo, Qiming Ai, Shangchen Zhou, and Chen Change Loy. Flexible piecewise curves estimation for photo enhancement. *arXiv preprint arXiv:2010.13412*, 2020.

[30] Jinxiu Liang, Yong Xu, Yuhui Quan, Jingwen Wang, Haibin Ling, and Hui Ji. Deep bilateral retinex for low-light image enhancement. *arXiv preprint arXiv:2007.02018*, 2020.

[31] Enyu Liu, Songnan Li, and Shan Liu. Color enhancement using global parameters and local features learning. In *Asian Conference on Computer Vision (ACCV)*, 2020.

[32] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 212–220, 2017.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[34] Kin Gwn Lore, Adedotun Akintayo, and Soumik Sarkar. Llnet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition (PR)*, 61:650–662, 2017.

[35] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579–2605, 2008.

[37] Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. Deeplpf: Deep local parametric filters for image enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12826–12835, 2020.

[38] Sean Moran, Steven McDonagh, and Gregory Slabaugh. Curl: Neural curve layers for global image enhancement. In *International Conference on Pattern Recognition (ICPR)*, pages 9796–9803. IEEE, 2021.

[39] Jongchan Park, Joon-Young Lee, Donggeun Yoo, and In So Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5928–5936, 2018.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:8026–8037, 2019.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.

[42] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, 2019.

[43] Li Tao, Chuang Zhu, Guoqing Xiang, Yuan Li, Huizhu Jia, and Xiaodong Xie. Llcnn: A convolutional neural network for low-light image enhancement. In *IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[44] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *ACM International Conference on Multimedia (ACM MM)*, pages 1041–1049, 2017.

[45] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5265–5274, 2018.

[46] Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6849–6857, 2019.

[47] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. *British Machine Vision Conference (BMVC)*, 2018.

[48] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for real image restoration and enhancement. In *European Conference on Computer Vision (ECCV)*, pages 492–511. Springer, 2020.

[49] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[50] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations (ICLR)*, 2018.

[51] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[52] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[53] Yonghua Zhang, Jiawan Zhang, and Xiaojie Guo. Kindling the darkness: A practical low-light image enhancer. In *ACM International Conference on Multimedia (ACM MM)*, pages 1632–1640, 2019.