NamedCurves: Learned Image Enhancement via Color Naming

David Serrano-Lozano[1,2], Luis Herranz[3], Michael S. Brown[4], and
Javier Vazquez-Corral[1,2]

[1] Computer Vision Center, Barcelona, Spain
[2] Universitat Autònoma de Barcelona, Barcelona, Spain
[3] Universidad Autónoma de Madrid, Madrid, Spain
[4] York University, Toronto, Canada

{dserrano,jvazquez}@cvc.uab.cat, luis.herranz@uam.es, mbrown@eecs.yorku.ca
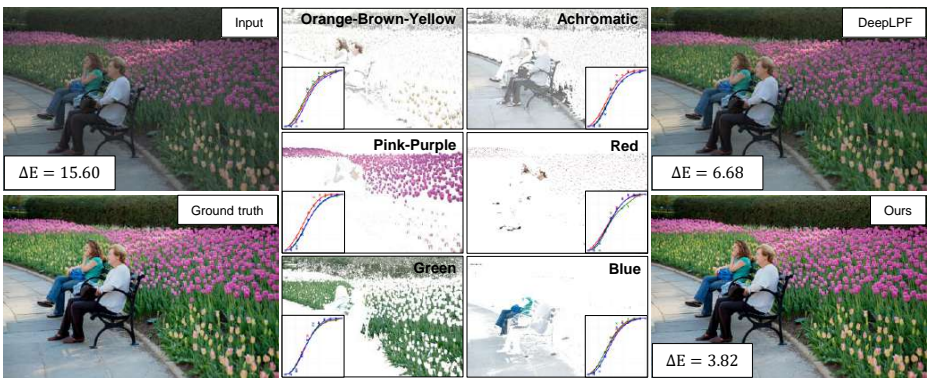
namedcurves.github.io

Fig. 1: Column 1 displays an input image corrected by a photo-editing expert (denoted as *ground truth*). Our proposed method decomposes the image based on color naming and learns a tone-curve correction to mimic the expert's style (shown in columns 2-3). Results comparing the input, our results, and the approach by [21] are reported in terms of the color distance metric ΔE_{00}.

Abstract. A popular method for enhancing images involves learning the style of a professional photo editor using pairs of training images comprised of the original input with the editor-enhanced version. When manipulating images, many editing tools offer a feature that allows the user to manipulate a limited selection of familiar colors. Editing by color name allows easy adjustment of elements like the "blue" of the sky or the "green" of trees. Inspired by this approach to color manipulation, we propose NamedCurves, a learning-based image enhancement technique that separates the image into a small set of named colors. Our method learns to globally adjust the image for each specific named color via tone curves and then combines the images using an attention-based fusion mechanism to mimic spatial editing. We demonstrate the effectiveness of our method against several competing methods on the well-known Adobe 5K dataset and the PPR10K dataset, showing notable improvements.

Keywords: Color enhancement · Image enhancement · Color naming

1   Introduction

Color plays a vital role in photography, enhancing focal points, evoking emotions, and enriching storytelling. Whether through vibrant hues or subtle tones, understanding the importance of colors is crucial for photographers seeking to evoke specific responses. Despite significant advancements in camera technology, amateurs and professionals still often resort to post-capture image enhancement to enhance an image's quality. However, manual enhancement can be challenging for those lacking expertise, time, or a well-developed aesthetic sense.

A potential solution to avoid manual adjustment is to learn a deep network model that can mimic the image editing style of a skilled photographer or colorist. These methods leverage a dataset of image pairs with the original and corresponding artist-edited images. It is interesting to consider the tools provided to the artists for performing the image editing. Many photo editing software applications (e.g., Adobe Photoshop [1]) provide users with the ability to manipulate the image based on a small set of fixed colors (e.g., red, green, yellow, orange, blue, purple). Interestingly, the predefined colors selected by software tools are similar to those linguists have found to be universal across languages [6], a research topic often referred to as *color naming*.

Contribution: We propose to leverage the use of color naming decomposition for image enhancement. In particular, we introduce NamedCurves, a learning-based image enhancement method that decomposes images into color names and estimates a tone curve in the form of smooth, differential Bezier curves (see Figure 1). This is followed by an attention-based fusion scheme that combines the images modified by the individual color curves, simulating local image editing. We compare our method with several state-of-the-art image enhancement methods on the MIT-Adobe-5K and PPR10K datasets. Our color naming scheme outperforms competing methods in terms of PSNR and ΔE.

2   Related Work

Related works are discussed for color naming and data-driven image-based enhancement methods that model professional editing styles.

2.1   Color Naming

Color naming is crucial for product designing, photography, and vision research [3, 24, 28, 35]. Berlin and Kay [6] conducted a study on the basic color lexicon across various languages and discovered universal semantics. Their seminal analysis showed that the evolution of basic color vocabularies is influenced by visual physiology, which limits the possible composite categories to a small number of those. The 11 color names found that most societies and cultures share are: *orange, brown, yellow, white, grey, black, pink, purple, red, green* and *blue*.
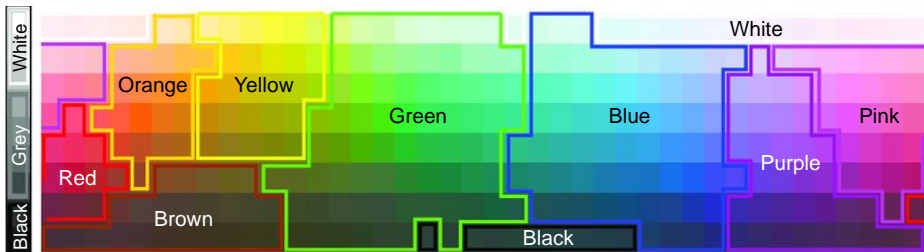
Fig. 2: Van de Weijer et al. [31] color names grouped in the Munsell color array. The color names are *orange, brown, yellow, white, grey, black, pink, purple, red, green* and *blue.*
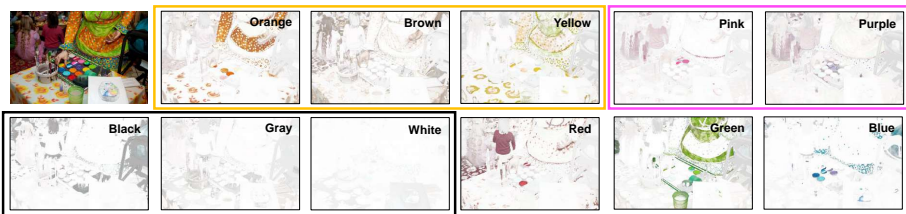

Fig. 3: Van de Weijer et al. [31] color naming method applied pixel-wise to the top-left image. The other 11 images show the 11 probability color names maps. Each color is displayed with a different map to aid visualization. Note that some linguistic color names share approximately the same hue and only differ in intensity— e.g., *pink* and *purple*. As tone curves are defined for all the intensity range we group: *orange-brown-yellow, pink-purple,* and *white-grey-black.* This grouping is represented by the boxes.

Following Berlin and Kay's research, different studies (e.g., [4, 31, 38]) aimed at predicting the boundaries between each of the color names. For example, Figure 2 shows the standard Munsell color array using Van de Weijer et al. [31] color classification based on color naming.

These methods work in the following manner. Given an RGB value in the sRGB color space, color naming methods produce an 11-d vector that corresponds to the probability of the RGB value belonging to each of the specific color names listed above. This is visualized in Figure 3, where we show an original image and the 11-probability maps coded with a color map to aid visualization. As described in Section 3.2, our method leverages the Van de Weijer et al. [31] color naming strategy to decompose each image in basic colors. However, we combine colors with similar hues (e.g., brown and orange) resulting in six color maps. In Figure 3 the colors grouped are shown in boxes.

2.2 Learned Image Enhancement

The need to provide users with tools to allow easy image enhancement has grown significantly due to the ease of photo-taking with smartphones. Initially,

histogram equalization was a primary method for enhancing contrast in images [25,27]. Subsequently, local operators [2,11] and color correction techniques based on color constancy [30] were introduced. Since the introduction of the MIT-Adobe-5K dataset by Bychkovsky et al. [7], which contains 5,000 images retouched by 5 experts, data-driven methods have emerged as one of the preferred means to improve image quality.

One category of these data-driven methods involves estimating intermediate or physical parameters for image retouching. Guo et al. [13] proposed Zero-DCE, the first method to formulate low-light image enhancement as a curve estimation problem. Their deep network estimates pixel-wise curves to modify the dynamic range of input images. This groundbreaking work influenced subsequent methods, such as CURL [22], which estimates piecewise linear curves for HSV, RGB, and CIELab color spaces; FlexiCurve [17], which estimates sets of piecewise curves and blends them via a Transformer and LTMNet [41] that learns a grid of tone curves to locally enhance an image. Additionally, Moran et al. [21] introduced DeepLPF, inspired by Photoshop's local filters tool, which estimates elliptical, gradual, and polynomial filters for local image editing. Wang et al. [32] proposed an alternative approach, estimating intermediate illumination maps for under-exposed images instead of directly learning image-to-image mappings.

Lookup tables (LUTs) represent another widely used method for image manipulation, typically manually tuned and fixed in camera imaging pipelines or photo editing tools. Zeng et al. [39] proposed 3DLUT, a method to learn these 3D LUTs from annotated data with a small convolutional neural network. Building upon this, Yang et al. [36] proposed AdaInt, a mechanism to achieve a more flexible sampling by learning the non-uniform sampling intervals. Wang et al. [33] also presented a modification of 3DLUT that incorporates spatial information to compute the image transformation.

Conversely, image-to-image methods estimate directly a mapping to modify the input images without intermediate steps. Generative adversarial networks (GANs) are frequently employed for such tasks. Chen et al. [8], Ni et al. [23], and Jiang et al. [15] proposed unpaired learning schemes using single GANs to estimate enhanced versions of input images directly.

As in previous methods [17,22,41], we use tone curves to manipulate images. However, we propose to leverage the use of color naming decomposition and an attention-based fusion scheme to mimic the image editing style of an expert.

3   Proposed Method

Figure 4 shows an overview of our proposed method, NamedCurves. Our method aims to enhance a low-quality RGB input image x, by a learned model that outputs an enhanced version $\hat{y}$. This enhanced image is derived as close as possible to the expert-retouched image y, based on some objective function L.

Our method consists of four main components, which are detailed in the following sections, including the loss function used for optimizing the framework. The approach first applies a DNN backbone that standardizes the input image
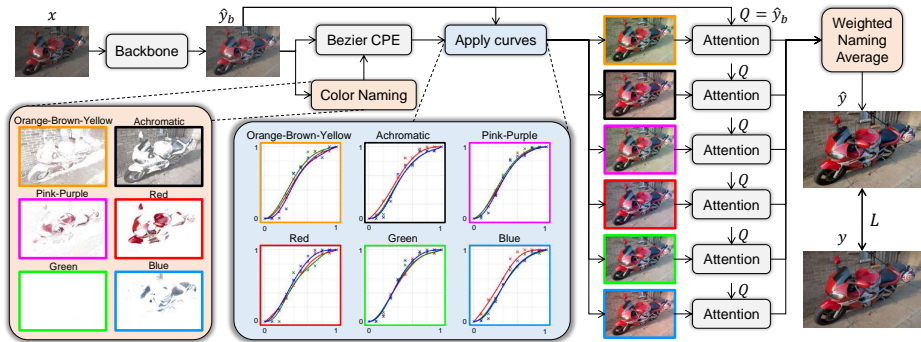
Fig. 4: Overview of the proposed method, NamedCurves. Our method aims to enhance an input image x. First, we use a UNet-like backbone to standardize the input image into a canonical latent space. Next, we decompose the standardized image $\hat{y}_b$ into six color probability maps (shown color-coded in the figure to aid visualization). Next, we learn a set of Bezier curves for each color name to manipulate the standardized image $\hat{y}_b$, obtaining six distinct globally adjusted images. Finally, an attention mechanism is used to fuse the edited images using as *Query* the standardized image $\hat{y}_b$ and as *Key* and *Value* the corresponding edited image. Our learning-based method uses an objective function L to compare the expert-retouched image with our final result $\hat{y}$.

into a canonical latent space. Next, we use color naming to decompose the image into six color maps. After color naming decomposition, a neural network learns a set of Bezier tone curves to manipulate each color map globally. Finally, an attention mechanism combines the edited images to achieve local editing effects.

3.1   Backbone

One challenge faced by learning-based image enhancement methods is that input images, x, can be captured using different cameras with different settings and under different lighting conditions. This may impact our ability for consistent color naming. Similar to the method by Moran et al. [21], we use a UNet-like backbone to standardize the input images.

Our backbone is inspired by the LPIENet [10] architecture. We use MobileNet layers (`Conv-DWConv-eLU`) [14] and a CBAM module [34]—a combination of spatial and channel attention. The backbone consists of three encoder blocks and two decoder blocks connected by multi-resolution skip connections. Each encoder block consists of the following: two MobileNet layers, a CBAM attention block, and a max-pooling layer. The decoder blocks follow the same structure except for the max-pooling layers that are replaced by bilinear upsampling layers. The multi-resolution skip connections consist of three parallel branches of convolutional layers with different dilation rates. Two of the paths consist of two `Conv-LeakyReLU` blocks to extract local information, while the other path consists of three `Conv-LeakyReLU-MaxPooling` blocks, an `AveragePooling` and a `LinearLayer` to extract global information. As in [12,20,22], we found that skip

connections at different resolutions improve the performance against backbones with simple skip connections.

3.2   Color Naming

We aim to decompose the standardized image $\hat{y}_b$ into a set of likelihood colors maps to focus different branches of the model. Due to the importance of memory colors—the green of the grass or the blue of the sky— in aesthetics [26, 29] we decided to use Color Naming, a perceptually-based color decomposition.

   We used the color naming model from Van de Weijer et al. [31] to obtain the probability maps for each color name. This model inputs an sRGB color value and outputs the probability of this color to belong to each of the 11 color naming categories, namely *red, blue, green, yellow, pink, purple, orange, brown, white, grey, black*. When applied to an image, the model operates for each pixel, which returns a set of probability maps.

   We note that some linguistic color names share similar hues, but only differ in intensity. For example, orange and brown or pink and purple. As tone curves are defined for all the intensity ranges, it will be beneficial to group these colors together. To this end, we reduce the set of 11 probability maps to just 6 by grouping *orange-brown-yellow*, *pink-purple*, and *white-grey-black* (referring to this last one as *achromatic*). The combined map for these cases is just the addition of the individual maps, and therefore, they are still probabilities (the sum of all the maps for a specific pixel is 1)—see supplementary for further details.

   Figure 4 shows the assignment of an input image to the six color maps. Note that the images have been color-coded to help visualize their probabilities, however, the colors associated with these maps are the original RGB values from the input (see supplemental materials). In the following section, we describe how each color map conditions a different set of RGB tone curves.

3.3   Bezier Curve Estimation

Similar to prior works [13, 17, 22], we leverage tone curves to remap the shadows, midtones, and highlights of the image $\hat{y}_b$ conditioned by the color naming probability maps. Tone curves represent global adjustments of the intensity from the input level to the output level. The curves are applied pixel-wise in each color channel as 1D Look-Up Tables. Bennet and Finlayson [5] demonstrated that tone adjustments are typically simple curves for a large dataset of enhanced images or can be well-approximated as such. We use Bezier curve parametrization to generate smooth and continuous tone curves from discrete control points.

   Specifically, we aim to estimate one global curve for each RGB channel c and color name n. Each curve is parameterized by M control points. We evenly distribute these control points along the input axis, with the first control point fixed at $(0, 0)$. Consequently, we only need to estimate the control points' output axis values instead of the two point coordinates, resulting in $M-1$ parameters
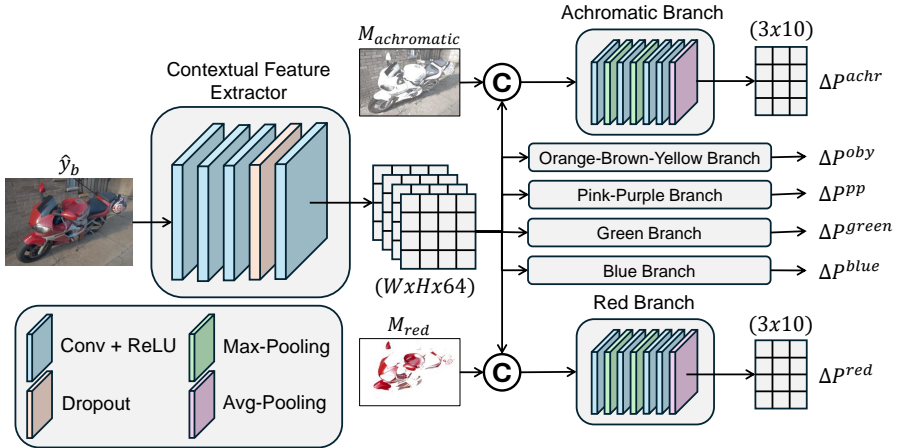
Fig. 5: Bezier Control Point Estimator (BCPE). First, we extract 64-D convolutional feature maps from $\hat{y}_b$ using mainly 4 `Conv-ReLU` blocks. Then, the color naming is concatenated and passed through 4 `Conv-ReLU-MaxPooling` and a final `AveragePooling` layer. The output of the module is ΔP, the unnormalized control points increments.

per curve. Thus, the Bezier formulation $B^{n,c}$ of a curve can be expressed as:

$$B^{n,c}(i) = \sum_{m=0}^{M-1} P_m^{n,c} \binom{M-1}{m} (1-i)^{(M-1-m)} i^m, \qquad (1)$$

where $i \in [0,1]$ is an image channel pixel, $P_m^{n,c}$ denotes the m-th control point for the color name n and channel c, and M is the total number of control points.

The Bezier Control Points Estimator (BCPE) aims to estimate the control points defining the Bezier tonal curves of an image. Figure 5 illustrates the BCPE, comprising two distinct blocks: the contextual feature extractor and 6 color naming branches. The contextual feature extractor primarily consists of 4 `Conv-ReLU` blocks and a `Dropout`, while the color naming branches consist of 4 `Conv-ReLU-MaxPooling` blocks and a final `AveragePooling` layer to manage the variable sizes of the images.

The contextual feature extractor computes a 64-d convolutional feature map from the standardized image $\hat{y}_b$. Each color naming branch receives as input a concatenation of the 64-d feature maps and the corresponding color naming probability map. The output of the branch for color name n consists of three sets of M increments $(\Delta P_m^{n,c})_{m=1}^{M}$, each set corresponding to a curve for a given color channel c. These increments $\Delta P_m^{n,c}$ do not directly represent the control points as we impose two different constraints. First, to make the curves monotonically increasing functions, we define $\Delta P_m^{n,c}$ as positive increments relative to the previous point. Second, we normalize $\Delta P_m^{n,c}$, ensuring the total increment between the first and last points is 1 and, thus, placing the last point at $(1,1)$.
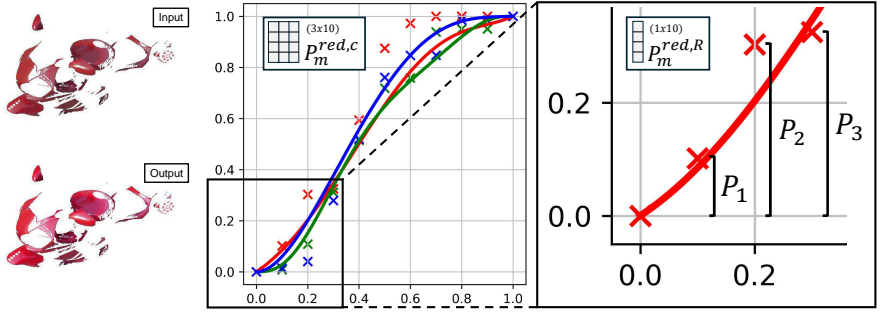
Fig. 6: An example of a Bezier curve (Red branch). The first column shows the input and edited image pixels in which the red color name branch focuses. The center plot shows the tone RGB curves learned for the red color name, while the right plot displays a zoomed-in view of the first four control points of the red channel.

Consequently, we compute the control points $P_m^{n,c}$ as the accumulated sum of the normalized $\Delta P_m^{n,c}$. This can be formulated as:

$$P_m^{n,c} = \frac{1}{S^{n,c}} \sum_{k=1}^{m} \Delta P_k^{n,c}, \tag{2}$$

where $S^{n,c} = \sum_{k=1}^{M} \Delta P_k^{n,c}$.

Figure 6 illustrates an example of a Bezier curve. The left column shows the input and output image pixels with higher red color name probability than 0.2 The center plot shows the tone curves learned for the red color name, while the right plot displays a zoomed-in view of the first four control points of the red channel. Note how the control points are fixed and evenly distributed along the input axis, while $P_m^{n,c}$ define the output axis value and, thus, the curvature of the tonal curve. The six sets of Bezier curves learned are applied pixel-wise to the entire standardized image $\hat{y}_b$, yielding six distinct globally-adjusted images.

3.4   Attention-based image fusion

Tone curves allow global color manipulation but cannot mimic local manipulation from experts in the training images. Here, we detail how to fuse these images via an attention-based mechanism that models their spatial dependencies.

For each of the 6 color name processed images, we use an attention module to compute a blending weight per image. Specifically, we employ $\hat{y}_b$ as the *Query*, while the corresponding globally-adjusted image serves as the *Key* and *Value*. Initially, we process the input images through 2 `Conv-ReLU-MaxPooling` blocks, yielding 16 convolutional feature maps with a broad receptive field. In particular, $Q, K, V \in \mathbb{R}^{W/8 \times H/8 \times 16}$. Finally, we aggregate $\hat{y}_b$ using two `Conv-ReLU-Upsampling` blocks. After the attention mechanism, we aggregate $\hat{y}_b$ via two

`Conv-ReLU-Upsampling` blocks. Note that there is a trade-off between the attention resolution and the computational cost. Downsampling the image by 8 did not introduce noticeable artifacts on the final image $\hat{y}$ after upsampling.

Ultimately, to generate the final image $\hat{y}$, we compute a weighted average of the obtained globally- and locally-adjusted images employing the original color naming probability maps. To mitigate potential artifacts arising from low probability values, we threshold the probability maps at 0.2, setting smaller values to 0. Subsequently, we normalize the maps to ensure each pixel sums to unity before performing the weighted average.

3.5 Loss Function

Our training loss function comprises three terms. The initial term calculates the L_2 loss between the standardized image $\hat{y}_b$ and the ground truth y, with a weighting factor α. The subsequent two terms compute the L_2 and $SSIM$ losses between the output of our model $\hat{y}$ and y. The primary objective of the first term is to obtain a good enough standardized output by the backbone. The other two terms are designed to assess the fidelity of the final output. In our experiments, we set α to 0.5 (see supplementary for an ablation study on this term). This value was determined to yield optimal performance and allows the image $\hat{y}_b$ to represent the scene colors accurately. The training loss is defined as:

$$L(\hat{y}_b, \hat{y}, y) = \alpha||y - \hat{y}_b||_2 + ||y - \hat{y}||_2 + (1 - SSIM(y, \hat{y})).$$ (3)

4 Experimental Results

4.1 Experimental setup

Datasets: We compare our method with state-of-the-art (SOTA) methods using the widely used MIT-Adobe-5K dataset [7] and the PPR10K dataset [19]. MIT5K consists of 5000 images captured independently using several DSLR cameras and retouched by five artists. However, although the image content is the same, different "versions" have emerged due to variations in image pre-processing and the number of training images. To make a fair comparison with all the SOTA methods, we have used three versions: (1) DPE [8], (2) UPE [32], and (3) UEGAN [23]. Specifically, DPE splits the data into 2250, 2250, and 500 images for training, validation, and testing, respectively. The last two versions split the images into 4500 images for training and 500 images for testing. DPE and UEGAN pre-process the images in the same manner but with different image sizes, while UPE pre-processes the input images to be under-exposed. Following [8, 17, 21, 22, 32] we only use the Expert-C retouched images as ground truth. PPR10K is a portrait photo retouching dataset with 11616 high-quality images retouched by 3 experts independently. We use the official splits, dividing the images into 8875 and 2286 for training and testing, respectively. Following [36] we conducted the experiments on the 360p augmented version which every image pair has 5 extra input versions with different manual settings. As in [19, 36, 39] we evaluate our method on the three expert-retouched versions of PPR10K.

Table 1: Quantitative comparisons on the DPE, UPE, and UEGAN versions of the MIT-Adobe-5K dataset. "-" means the source code or models are unavailable, or results for the corresponding metric were not in the original paper.

MIT-5K	Method	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓	Time (ms)
	DPE [8]	23.80	0.900	-	-	-	-
	DeepLPF [21]	23.93	0.903	0.040	7.00	8.05	136
DPE	CURL [22]	24.04	0.900	-	-	-	102
	FlexiCurve [17]	24.37	0.920	0.060	-	-	-
	NamedCurves (Ours)	**24.91**	**0.927**	**0.038**	**6.60**	**7.82**	26
	DPE [8]	22.15	0.850	0.108	-	-	-
	UPE [32]	23.04	0.893	0.158	-	-	-
UPE	LTMNet [41]	24.27	**0.913**	0.068	-	-	-
	DeepLPF [21]	24.48	0.887	0.103	6.89	7.77	136
	NamedCurves (Ours)	**25.20**	0.906	**0.047**	**6.54**	**7.58**	26
	InstructIR [9]	24.65	0.900	-	-	7.61	-
	3DLUT [39]	25.29	0.923	0.043	6.76	7.55	13
UEGAN	SepLUT [37]	25.47	0.921	0.042	6.71	7.49	10
	AdaInt [36]	25.49	0.926	0.041	6.69	7.47	13
	NamedCurves (Ours)	**25.59**	**0.936**	**0.038**	**6.07**	**7.40**	26

Implementation Details: We trained our model using Adam [16], an initial learning rate 1e-4, reduced by 50% every 50 epochs. We use horizontal flips for augmenting the training data. We chose the model with the best validation ΔE_{00} in the DPE version of the MIT5K, while we trained for a fixed 200 and 100 epochs for the other versions of MIT5K and PPR10K, respectively.

4.2   Comparison with SOTA Methods

We compare our method with SOTA methods using the MIT5K and the PPR10K datasets, using the corresponding evaluation metrics. Specifically, in the MIT5K comparisons, we used PSNR, SSIM, LPIPS [40], ΔE_{00} and ΔE_{ab}. MIT5K results are presented in Table 1. We also report the inference time for a 480p image on an AMD EPYC 7642 and a single NVIDIA A40. Our full architecture outperforms contemporary curve estimation methods; CURL [22], FlexiCurve [17] and LTMNet [41] on the DPE and UPE versions of MIT5K. Similarly, our method outperforms all-in-one, image-to-image, and LUT-based methods across all the versions. Table 2 shows results on the PPR10K dataset. Following [19, 36], we used PSNR and ΔE_{ab} to compare our method with the state-of-the-art. As the other methods did not compute SSIM, LPIPS and ΔE_{00} and the pre-trained models are unavailable, we report these metrics for our model in the supplementary material. We outperform all the contemporary methods on both PSNR and ΔE_{ab} on the three expert versions of the dataset.

Figure 7 provides several qualitative comparisons, showing examples from both the MIT5K and the PPR10K datasets in Figure 7a and 7b, respectively. Our
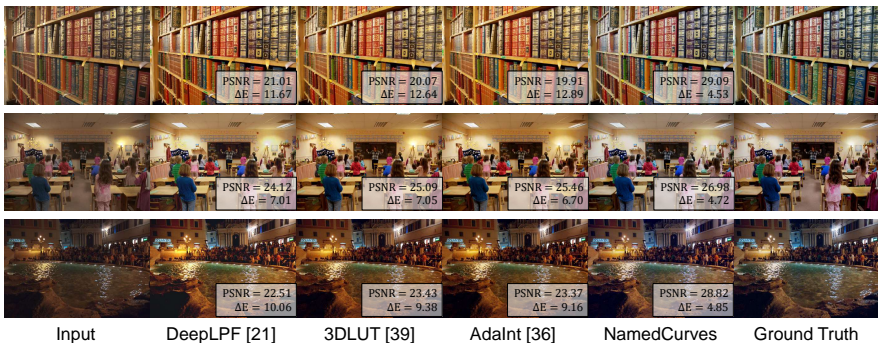
(a) MIT-5K dataset qualitative results.

(b) PPR10K dataset qualitative results.

Fig. 7: Qualitative comparisons on the MIT-5K (Figure 7a) and PPR10K datasets (Figure 7b). On the bottom-right of each image we display the PSNR and ΔE_{00}.

Table 2: Quantitative comparisons the PPR10K dataset. We only compute mean PSNR and ΔE_{ab} since most other methods do not have models available for inference.

PPR10K	Expert A		Expert B		Expert C	
Method	PSNR $\uparrow$	$\Delta E_{ab} \downarrow$	PSNR $\uparrow$	$\Delta E_{ab} \downarrow$	PSNR $\uparrow$	$\Delta E_{ab} \downarrow$
HDRNet [18]	23.93	8.70	23.96	8.84	24.08	8.87
3DLUT [39]	25.64	6.97	24.70	7.71	25.18	7.58
SepLUT [37]	26.28	6.59	25.23	7.49	25.59	7.51
AdaInt [36]	26.33	6.56	25.40	7.33	25.68	7.31
NamedCurves (Ours)	**26.81**	**6.48**	**25.91**	**7.18**	**25.69**	**7.27**

method provides visually appealing results that resemble the expert-retouched version regarding color fidelity. In Figure 7a, for instance, our method accurately rectifies color casts in the first two rows. Lastly, in the third row, our method demonstrates superior performance in enhancing nighttime images. Similarly, in Figure 7b our method outperforms AdaInt [36] in replicating the expert-

Table 3: Ablation studies on the contributions of each module of our proposed method.

#	Backbone	Curves	CN	Att.	WN-Avg	PSNR ↑	SSIM ↑	ΔE_{00} ↓
1		6	✓	✓	✓	23.40	0.912	8.87
2	✓					23.74	0.916	8.68
3	✓	1				24.09	0.921	7.53
4	✓	6				24.24	0.922	7.50
5	✓	6	✓			24.56	0.926	7.07
6	✓	6	✓		✓	24.68	0.926	6.88
7	✓	6	✓	✓		24.60	0.926	6.92
8	✓	6	✓	✓	✓	**24.91**	**0.927**	**6.60**

retouched image on the PPR10K-A dataset. This is particularly noticeable in regions such as the background brown wall (first row) and the global color temperature (last row).

We provide additional qualitative examples in Figure 8. The top row shows the outcomes generated by the methods, while the bottom row displays the per-pixel ΔE_{00} error maps. In the first row, the other methods struggle to address color cast issues effectively, leading to significant ΔE_{00} values across the entire image. Conversely, our method demonstrates superior performance, yielding minimal errors confined primarily to small regions. The second image has two distinct areas: one characterized by intricate details and the other by a plain surface. DeepLPF [21] exhibits artifacts, such as the elliptical distortion in the grey area. Similarly, 3DLUT [39] and AdaInt [36] show limitations by enhancing properly only one of the image regions, thereby resulting in substantial errors in the other segment. In contrast, our method consistently enhances the photograph across the entire image by seamlessly integrating both local and global adjustments.

User Study: We compared our method against AdaInt [36] and SepLUT [37] following a two-alternative forced choice (2AFC), performed in a completely black room with a monitor set to sRGB. We randomly selected 25 images from both MIT5K and PPR10K datasets. 15 observers took part, all tested for colorblindness with the Ishihara test. Results analyzed using Thurstone Case V (larger means better) were: NamedCurves: 1.12; AdaInt: -0.38; SepLUT: -0.74. Our method is statistically significantly better than the other two —95% confidence interval is 0.33.

4.3   Ablation studies

In this section, we choose the DPE version of MIT5K to conduct several ablation studies to verify the proposed method. We performed experiments to understand the effectiveness of the individual modules used by our framework. Table 3 presents the results for various combinations of the modules of our method. We report PSNR, SSIM, and ΔE_{00}. Throughout the experiments, we consistently
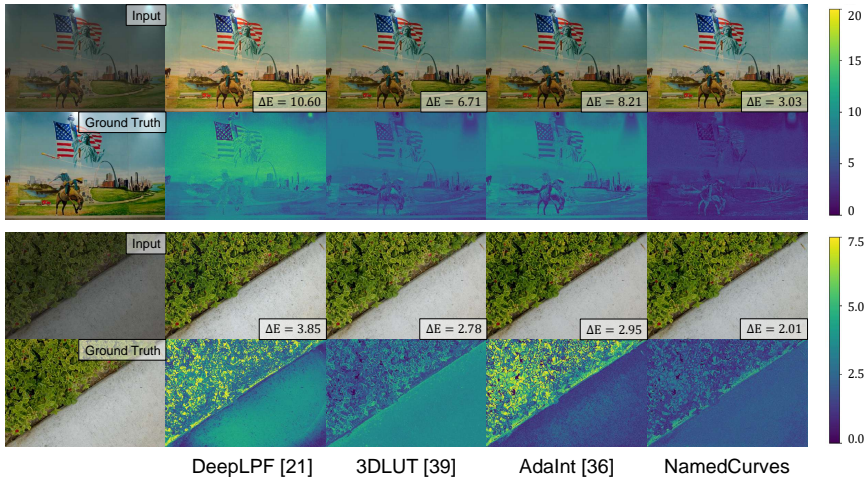
Fig. 8: Qualitative results of two images of the MIT-Adobe-5K. The first column presents the input and the expert-retouched image. The top row shows the estimated enhanced version of the input image of DeepLPF [21], 3DLUT [39], AdaInt [36] and our method. Under each image, we present the ΔE_{00} error map. On the bottom-right of each image, we display the mean ΔE_{00}.

utilize the backbone while incrementally incorporating different modules of our method. The column labeled *Curves* indicates the number of RGB Bezier curves utilized and, consequently, the number of globally adjusted images produced. The *color naming* column (*CN*) specifies whether the color naming probability maps are concatenated with the 64-d feature maps used by the Beizer curve manipulation. Note that if we use the color naming probability maps we must use six curves. The Attention (*Att.*) column indicates whether we apply local modifications to the globally adjusted images before fusing them. Finally, the *WN-Avg* column denotes whether the experiment employs the color naming maps to weigh the images before blending them. In cases where we do not use WN-Avg and there are multiple output images, we simply average them.

This table shows how combining the different modules of our model improves the performance. Each of our additions improves the result. Experiment 1 emphasizes the importance of the backbone of our model. In detail, color naming modules produce the largest boosts in performance. We gain 0.32 dB in PSNR and 0.43 in ΔE_{00} when we concatenate the color naming maps to the Bezier Control Point Estimator feature maps - experiment 5. Furthermore, we also gain 0.31 dB in PSNR and 0.32 in ΔE_{00} when we use the color naming maps to weight the final average - experiment 8.

We further investigate assessing our backbone's impact on our model's performance. We evaluate by using different backbones from other methods. Table 4a reports the results of our experiments. Notably, our proposed backbone yields superior performance compared to other methods' backbones, namely UNet [21],

Table 4: Ablation studies for the (a) backbone architecture and (b) the number of control points in the Bezier curves.

Backbone	PSNR ↑	SSIM ↑	ΔE_{00} ↓
UNet [21]	24.49	0.920	7.04
LPIENet [10]	24.51	0.920	7.07
TED [22]	24.70	0.925	6.97
NamedCurves	**24.91**	**0.927**	**6.60**

(a) Ablation on backbone

N	PSNR ↑	SSIM ↑	ΔE_{00} ↓
5	24.69	0.924	6.82
7	24.88	0.926	6.76
11	**24.91**	**0.927**	**6.60**
16	24.52	0.920	6.85

(b) Ablation on number of control points

Fig. 9: Example of an image with just two dominant colors. Our method still outperforms the others, but does not obtain the same level of advantage.

TED [22] and LPIENet [10]. Importantly, irrespective of the utilized backbone, our method consistently outperforms the previous state-of-the-art models on this DPE version of MIT5K. This highlights the significance of leveraging color naming as the main contributing factor to the performance of our method.

We perform a final ablation study on the number of control points N. We tested our method with 5, 7, 11, and 16 control points for each Bezier curve. In Table 4b we report the PSNR, SSIM, and ΔE_{00} of every experiment. We found that 11 control points –every 0.1 in the input axis– works best for our method.

4.4   Limitations

Our method aims to replicate the image style of a skilled photographer estimating a series of tone curves for each color name. However, our method loses part of its advantage compared to the previous SOTA methods in scenarios where the image comprises few color regions. In such instances, different branches of the method receive low-weighting values, resulting in an almost global adjustment technique (e.g. image dominated by just two color names, see Figure 9).

5   Conclusion

This paper introduced a new image enhancement model based on color naming that outperforms the current state-of-the-art across various versions of the Adobe 5K and the PPR10K datasets. Our approach uses expert-edited images for learning and explicitly separates the image into a small set of named colors. It learns to adjust the image for each specific named color and then combines the images using an attention-based fusion mechanism.

Acknowledgements

References

1. Adobe Systems: Adobe photoshop. Computer software, photoshop.adobe.com 2
2. Aubry, M., Paris, S., Hasinoff, S.W., Kautz, J., Durand, F.: Fast local laplacian filters: Theory and applications. ACM TOG **33**(5), 1–14 (2014) 4
3. Bahng, H., Yoo, S., Cho, W., Park, D.K., Wu, Z., Ma, X., Choo, J.: Coloring with words: Guiding image colorization through text-based palette generation. In: ECCV. pp. 431–447 (2018) 2
4. Benavente, R., Vanrell, M., Baldrich, R.: Parametric fuzzy sets for automatic color naming. JOSA A **25**(10), 2582–2593 (2008) 3
5. Bennett, J., Finlayson, G.D.: Simplifying tone curves for image enhancement (2023) 6
6. Berlin, B., Kay, P.: Basic color terms: Their universality and evolution. Univ of California Press (1991) 2
7. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input/output image pairs. In: CVPR (2011) 4, 9
8. 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: CVPR (2018) 4, 9, 10
9. Conde, M.V., Geigle, G., Timofte, R.: High-quality image restoration following human instructions. arXiv preprint arXiv:2401.16468 (2024) 10
10. Conde, M.V., Vasluianu, F., Vazquez-Corral, J., Timofte, R.: Perceptual image enhancement for smartphone real-time applications. In: WACV. pp. 1848–1858 (2023) 5, 14
11. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. In: SIGGRAPH (2002) 4
12. Gharbi, M., Chen, J., Barron, J.T., Hasinoff, S.W., Durand, F.: Deep bilateral learning for real-time image enhancement. ACM TOG **36**(4), 1–12 (2017) 5
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: CVPR (2020) 4, 6
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 5
15. 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 TIP **30**, 2340–2349 (2021) 4

16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
17. Li, C., Guo, C., Zhou, S., Ai, Q., Feng, R., Loy, C.C.: Flexicurve: Flexible piecewise curves estimation for photo retouching. In: CVPRW (2023) 4, 6, 9, 10
18. Li, J., Fang, P.: Hdrnet: Single-image-based hdr reconstruction using channel attention cnn. In: ICMSSP (2019) 11
19. 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: CVPR (2021) 9, 10
20. Marnerides, D., Bashford-Rogers, T., Hatchett, J., Debattista, K.: Expandnet: A deep convolutional neural network for high dynamic range expansion from low dynamic range content. In: Comput. Graph. Forum. vol. 37, pp. 37–49 (2018) 5
21. Moran, S., Marza, P., McDonagh, S., Parisot, S., Slabaugh, G.: Deeplpf: Deep local parametric filters for image enhancement. In: CVPR (2020) 1, 4, 5, 9, 10, 12, 13, 14, 21, 23
22. Moran, S., McDonagh, S., Slabaugh, G.: Curl: Neural curve layers for global image enhancement. In: ICPR (2021) 4, 5, 6, 9, 10, 14, 21
23. Ni, Z., Yang, W., Wang, S., Ma, L., Kwong, S.: Towards unsupervised deep image enhancement with generative adversarial network. IEEE TIP **29**, 9140–9151 (2020) 4, 9
24. Regier, T., Kay, P., Khetarpal, N.: Color naming reflects optimal partitions of color space. Proceedings of the National Academy of Sciences **104**(4), 1436–1441 (2007) 2
25. Reza, A.M.: Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. J VLSI SIG PROC SYST **38**, 35–44 (2004) 4
26. Shing-Sheng, G., Po-Sung, H.: Influences of psychological factors on image color preferences evaluation. Color Research & Application **35**(3), 213–232 (2010) 6
27. Stark, J.A.: Adaptive image contrast enhancement using generalizations of histogram equalization. IEEE TIP **9**(5), 889–896 (2000) 4
28. Szafir, D.A.: Modeling color difference for visualization design. IEEE transactions on visualization and computer graphics **24**(1), 392–401 (2017) 2
29. Topfer, K., Cookingham, R.: The quantitative aspects of color rendering for memory colors. In: IS&T's PICS (2000) 6
30. Van De Weijer, J., Gevers, T., Gijsenij, A.: Edge-based color constancy. IEEE TIP **16**(9), 2207–2214 (2007) 4
31. Van De Weijer, J., Schmid, C., Verbeek, J., Larlus, D.: Learning color names for real-world applications. IEEE TIP **18**(7), 1512–1523 (2009) 3, 6, 18, 19, 20
32. Wang, R., Zhang, Q., Fu, C.W., Shen, X., Zheng, W.S., Jia, J.: Underexposed photo enhancement using deep illumination estimation. In: CVPR (2019) 4, 9, 10
33. 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: ICCV (2021) 4
34. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: ECCV. pp. 3–19 (2018) 5
35. Xue, D., Corral, J.V., Herranz, L., Zhang, Y., Brown, M.S.: Integrating high-level features for consistent palette-based multi-image recoloring. In: Comput. Graph. Forum. vol. 42, p. e14964. Wiley Online Library (2023) 2
36. Yang, C., Jin, M., Jia, X., Xu, Y., Chen, Y.: Adaint: Learning adaptive intervals for 3d lookup tables on real-time image enhancement. In: CVPR (2022) 4, 9, 10, 11, 12, 13, 21, 23, 24

37. Yang, C., Jin, M., Xu, Y., Zhang, R., Chen, Y., Liu, H.: Seplut: Separable image-adaptive lookup tables for real-time image enhancement. In: European Conference on Computer Vision. pp. 201–217. Springer (2022) 10, 11, 12
38. Yu, L., Cheng, Y., van de Weijer, J.: Weakly supervised domain-specific color naming based on attention. In: ICPR. IEEE (2018) 3
39. 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 TPAMI **44**(4), 2058–2073 (2020) 4, 9, 10, 11, 12, 13, 21, 23
40. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 10
41. Zhao, L., Abdelhamed, A., Brown, M.S.: Learning tone curves for local image enhancement. IEEE Access **10**, 60099–60113 (2022) 4, 10

Supplementary Material

The supplemental material provides additional information that could not be incorporated into the main paper due to page limit constraints. Specifically, we discuss (1) Adobe's color decomposition method, (2) more information on color naming probability maps, (3) further justifications on the color names grouping, (4) loss function parameter α ablation study, and (5) additional results.

Adobe Color Decomposition

Adobe Photoshop and Adobe Lightroom are software tools to allow photo editors the ability to fine-tune individual colors within an image. Our method is inspired by these tools. In particular, the software decomposes the image into a predefined set of colors (*red, orange, yellow, green, cyan, blue, purple* and *pink*), enabling users to independently manipulate the hue, saturation, and luminance of each color. In Figure 10, we show two screenshots of the tools and examples edited using this feature. For each example, we display the top of the input image alongside the default parameter values for the color to fine-tune. We show the edited image in the bottom images alongside the corresponding slider adjustments. In the left example, we demonstrate the modification of *blue*, illustrating alterations in the sky while preserving non-*blue* regions. In the right example, we focus on adjusting *purple*, a non-primary color. This allows us to selectively modify specific *purple* elements, such as the girl's clothing, without affecting the rest of the image. Notably, adjustments to the desired color sliders induce changes across all three color channels, as evident in the histograms provided in the top-right corner.

Color Naming Probability Maps

As discussed in the main paper, we use the color naming method proposed by Van de Weijer et al. [31]. This method is applied on each pixel. Given an sRGB image, this method generates 11 probability maps, each corresponding to a distinct color name: *red, blue, green, yellow, pink, purple, orange, brown, white, grey, black*. As discussed in the main paper, we group certain color names due to their similar hues, differing primarily in intensity only. Specifically, we merge *orange-brown-yellow*, *pink-purple*, and *white-grey-black* (referring to this last one as *achromatic*). The combined maps are obtained by summing the individual probability maps. In the end, we obtain probability maps for 6 color categories.

Figure 11 illustrates these color-naming probability maps using different visualizations. In the first two examples, we depict the 11 color naming probability maps using the same color map for all the color names. In the third and fourth examples, we show the image pixels exceeding 0.2 probability for the 11 color names and the 6 color-category version, respectively. In this case, the pixels in the maps represent the real sRGB values in the original image. Finally, in the last example, we use the same visualization method as in the main submission (i.e., Figures 1, 3, and Figure 4). We also account for the probabilities assigned
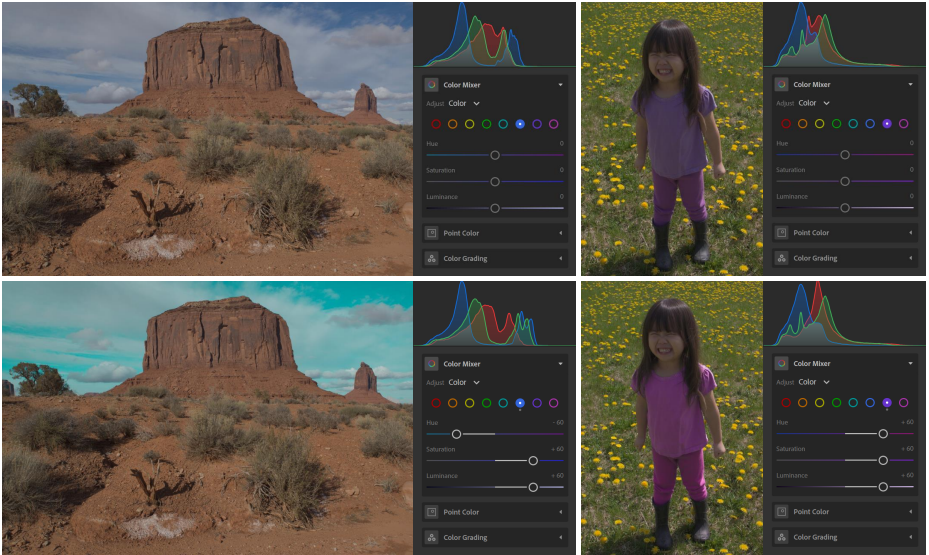
Fig. 10: Two examples of the Adobe Color Decomposition tool. In the left example, we manipulate the hue, saturation, and luminance of the color *blue*, while in the right example, we modify the color *purple*.

to each color to emphasize the probability aspect of color naming. In particular, for each pixel and color name, we compute:

$$n = (1 - p_i)I_w + p_iI_i, \tag{4}$$

where p_i represents the color name probability of pixel i, I_w denotes a white RGB value (i.e., [1, 1, 1]) and I_i signifies the RGB value of pixel i. It is important to note that the colors represented in these maps are not the original RGB values of the image, as they are scaled by the color naming probability p.

Color Names Grouping

We used the color naming model from Van de Weijer et al. [31] to obtain the probability map for each color name, namely *red, blue, green, yellow, pink, purple, orange, brown, white, grey, black*. However, we note that some linguistic color names share similar hues, but only differ in intensity. As tone curves are defined for all the intensity ranges, it will be beneficial to group these colors together. To this end, we reduce the set of 11 probability maps to just 6 by grouping *orange-brown-yellow*, *pink-purple*, and *white-grey-black* (referring to this last one as *achromatic*). In Figure 12, we visually show the reason for reducing the number of Color Naming channels to just 6. The figure illustrates 2D plots depicting the relationship between input and output intensity values for pixels with probability > 0.5 belonging to each specific color. For example, in the case
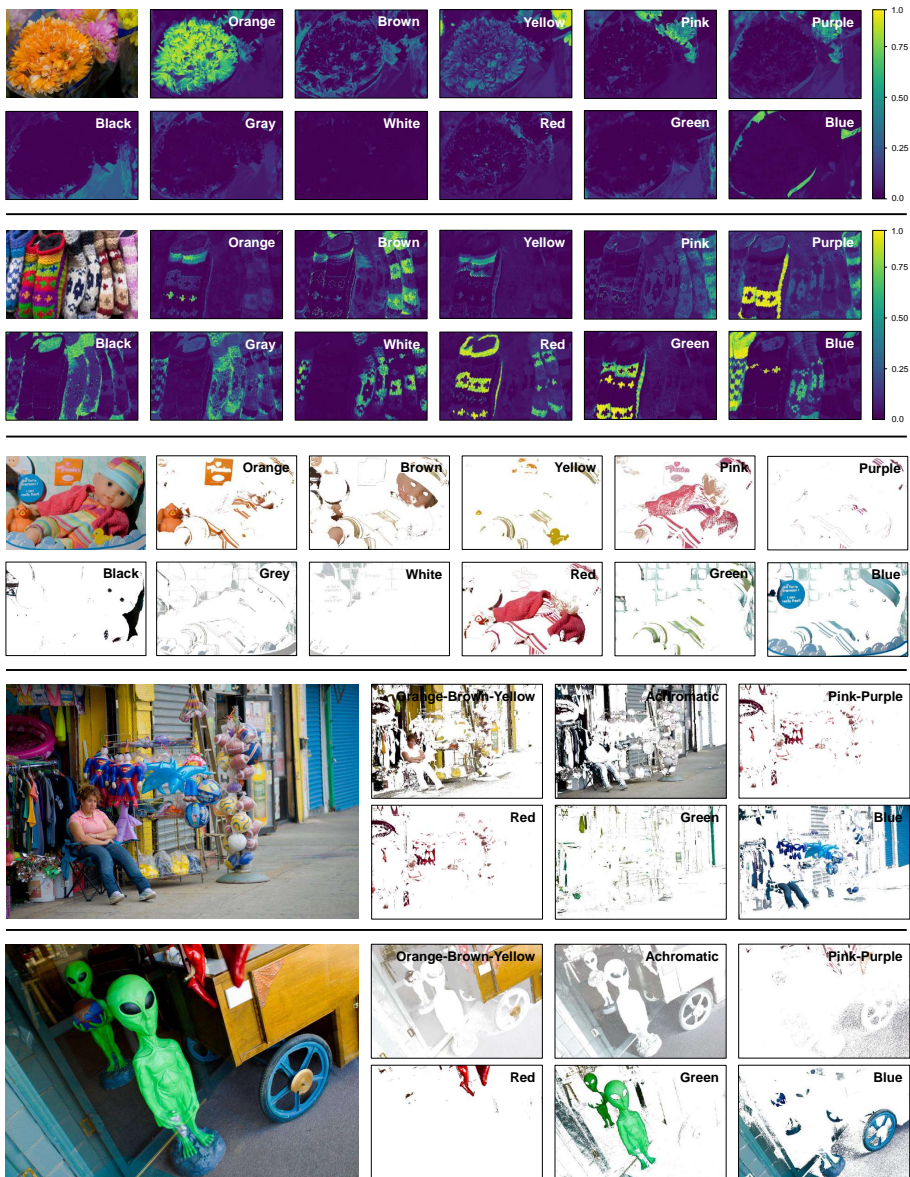
Fig. 11: Van de Weijer et al. [31] color naming method applied pixel-wise to five images. The color naming probability maps of the first two examples are displayed with the same color map. The third and fourth rows present visualization examples where we display the image pixels whose color naming map is higher than 0.2. The last row presents the same visualization as in the main paper, where we also take into consideration the probability value.
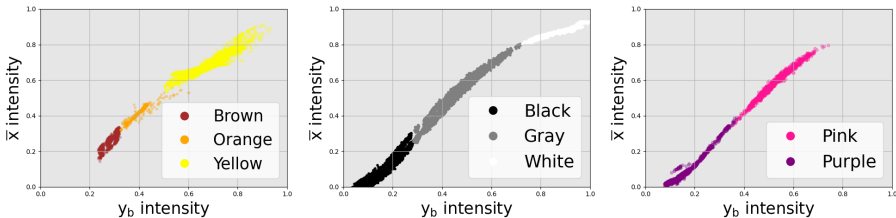
Fig. 12: Joined color names with respect to the intensity value. Pixels with probability > 0.5 are plotted.

Table 5: Ablation study on the α parameter of the loss function on the MIT5K-DPE dataset.

α	PSNR	ΔE_{00}
0	24.58	6.76
0.5	**24.91**	**6.60**
1	24.73	6.62

of orange-brown-yellow, brown is only present at low intensities, while orange dominates at mid-intensities and yellow at top intensities. The same analysis extends to the other joined color channels. Our method aims to learn a curve to be applied at all the intensity levels. Thus, incorporating information spanning all the intensity levels is beneficial. To also show this numerically, we experimented using our model with the 11 color terms, obtaining a PSNR of 24.72 dB (in comparison to 24.91 dB with 6 channels) in the MIT5K-DPE dataset.

Loss Function Ablation Study

Following prior works (DeepLPF [21], CURL [22]) our method starts with a backbone that serves to standardize the input. We consider three loss terms: the l_2 loss between the output of the backbone and the reference image and the l_2 and SSIM losses between the final output and the reference image. We chose to weight the backbone loss by $\alpha = 0.5$. An ablation study is shown in Table 5, where we can see that $\alpha = 0.5$ gives better results than $\alpha = 0$ (i.e., ignore the backbone output) and $\alpha = 1$ (i.e., heavily weight backbone output).

Additional MIT5K and PPR10K qualitative results

Figure 13 and Figure 14 show additional results from the MIT5K and PPR10K, respectively. We compare our method with DeepLPF [21], 3DLUT [39] and AdaInt [36].

Table 6: Additional quantitative results of our method on the PPR10K dataset. No other method computes these metrics in their respective papers for the PPR10K dataset.

Expert	SSIM ↑	LPIPS ↓	ΔE_{00} ↓
A	0.957	0.031	5.46
B	0.956	0.032	5.61
C	0.949	0.032	5.68

Additional PPR10K quantitative results

Table 6 reports SSIM, LPIPS and ΔE_{00} of our model on experts A, B and C of PPR10K. These metrics are not computed by other methods in their respective papers.
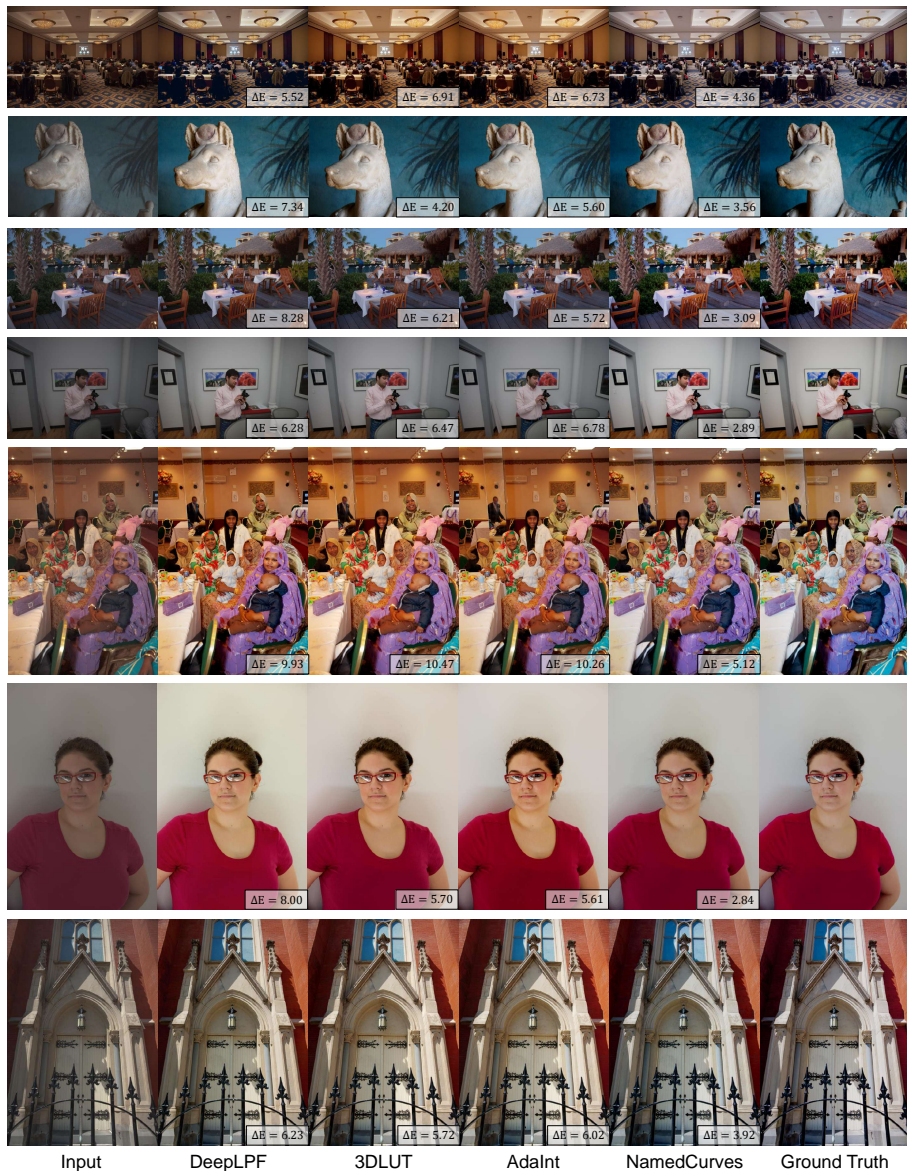
Fig. 13: Additional qualitative results on the MIT5K dataset. From left to right: the input image, DeepLPF [21], 3DLUT [39], AdaInt [36], our method, and the ground truth. ΔE_{00} is shown in the bottom-right corner of each image.
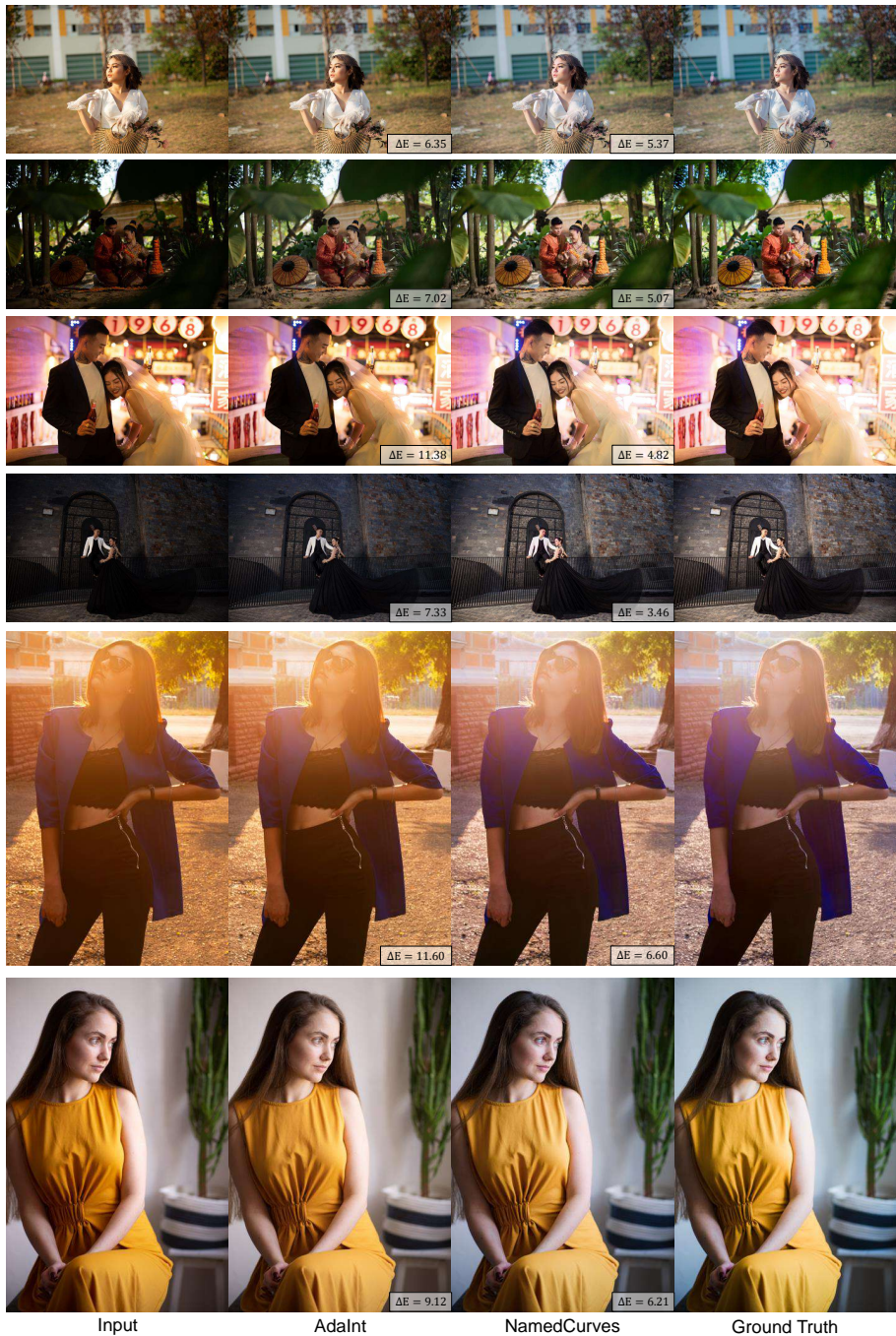
Fig. 14: Additional qualitative results performed using the PPR10K dataset. From left to right: the input image, AdaInt [36], our method, and the ground truth. ΔE_{00} is shown in the bottom-right corner of each image.