Fast Nonlinear Image Unblending

Daichi Horita[1]* Kiyoharu Aizawa[1] Ryohei Suzuki[2] Taizan Yonetsuji[2] Huachun Zhu[2]
[1]The University of Tokyo [2]Preferred Networks, Inc.

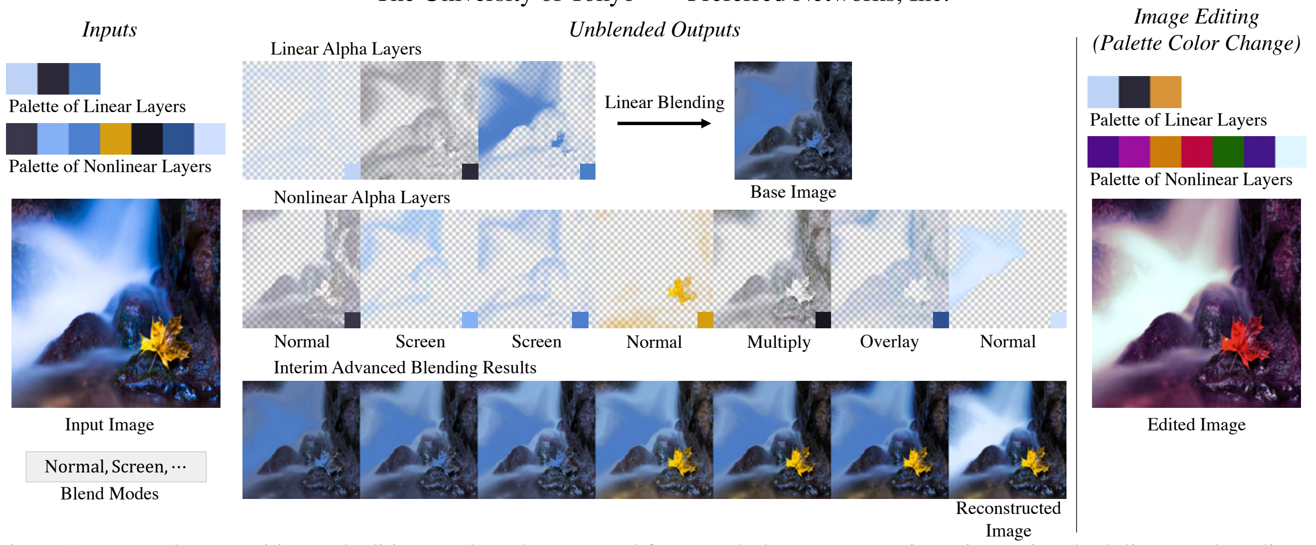


Figure 1. Image decomposition and editing results. The proposed framework decomposes an input image into both linear and nonlinear alpha layers based on specified color palettes and blend modes. In the bottom row of the middle column, we show the interim results of reconstructing the image using iterative advanced blending. After decomposition, we edit the image by changing the palette colors. Image courtesy of Theo Crazzolara.

Abstract

Nonlinear color blending, which is advanced blending indicated by blend modes such as "overlay" and "multiply," is extensively employed by digital creators to produce attractive visual effects. To enjoy such flexible editing modalities on existing bitmap images like photographs, however, creators need a fast nonlinear blending algorithm that decomposes an image into a set of semi-transparent layers. To address this issue, we propose a neural-network-based method for nonlinear decomposition of an input image into linear and nonlinear alpha layers that can be separately modified for editing purposes, based on the specified color palettes and blend modes. Experiments show that our proposed method achieves an inference speed 370 times faster than the state-of-the-art method of nonlinear image unblending, which uses computationally intensive iterative optimization. Furthermore, our reconstruction quality is higher or comparable than other methods, including linear blending models. In addition, we provide examples that apply our method to image editing with nonlinear blend modes.

*Part of this work was done while Daichi Horita was a Research Internship with Preferred Networks, Inc.

1. Introduction

Alpha blending is a procedure wherein the background colors in an image are mixed with the foreground colors. In particular, advanced blending (hereinafter referred to as nonlinear blending) [3, 4] comprises a total of 12 defined and four popular blend modes—*normal*, *multiply*, *screen*, and *overlay* —that are widely used by digital creators to produce attractive effects with great flexibility. For example, a creator can utilize the *screen* or *multiply* blend mode to produce the desired mix of lighting and darkening effects, as shown in Figure 2. Such nonlinear operations assume that images have layer structures. To perform nonlinear blending on arbitrary RGB images such as photographs or illustrations that do not have layer structures, we need to first decompose images into sets of semi-transparent layers with specific blend modes.

Soft segmentation is a particularly useful type of unblending operation in which an image is decomposed into a set of semi-transparent layers. Soft color segmentation methods [5, 7, 15, 17, 22, 24, 25] decompose images into several semi-transparent layers, each of which has a specific uniform color. However, most of these methods [5, 7, 17, 22, 24, 25] are limited to linear color unblending, i.e., a reconstructed image is the sum of the color palettes weighted by the corresponding decomposed alpha layers. Such linear unblending is not applicable to advanced blending of arbitrary RGB images. The first nonlinear color unblending method proposed by Koyama et al. [15] generalized the color unblending formulation to advanced blending operations. Since this method assumes an iterative optimization approach, the unblending operation is time consuming, which limits its practical use in cases that involve trial-and-error procedures to explore the combination of multiple blend modes.

In this paper, we propose the first neural-network-based nonlinear color unblending method, which decomposes an input image into its nonlinear alpha layers when given the blend modes and color palettes. Our proposed framework consists of five stages, as shown in Figure 3. First, we determine the palette colors by an automatic algorithm or by manual selection, which assigns a uniform color to each decomposed layer. The decomposed linear layers are then composited into a base image by linear blending, which is used as the background layer for advanced blending. Finally, the base image as a background layer is composited with the nonlinear as the foreground layers using advanced blending to reconstruct the original image.

As a typical use case of our method, an RGB image is first decomposed into soft color layers with automatically extracted palettes. Then, the palette colors and blend modes are specified according to particular use cases. In the experiments, we show that our method is approximately 370 times faster for inference than an existing method [15]. Specifically, our method decomposes a 2-megapixel (MP) image in 0.2 s on a GPU and empowers users to experiment with different combinations without waiting.

The primary contributions of this paper are as follows:

1. We propose the first neural-network-based method for decomposing an image into a set of linear and nonlinear semi-transparent layers according to a given color palette.

2. Through experiments, we show that the inference speed of the proposed method is 370 times that of the compared nonlinear method. Additionally, we observe that the reconstruction quality of our method is better than or comparable to those of previous methods, including linear blending models.

3. We provide several examples of image editing using the proposed method. Owing to the fast inference, various combinations of edits can be performed via trial-and-error procedures.

2. Related Works

2.1. Palette Representation

Palette extraction is a problem wherein a small number of representative colors are generated from an input image. There are two main approaches to extracting palettes. The first is a geometric approach that computes all the color samples and constructs the palette by calculating the convex hull [24, 25, 26]. For example, RGBXY proposed by Tan et al. [24] constructed the convex hull of an RGBXY-space for a linear blending model to obtain a color representation. The second approach uses clustering methods [5, 9, 13, 20, 22]. Chang et al. [9] formulated interactive image recoloring applications by editing extracted color palettes.

Several studies [7, 15] have demonstrated that using palettes with explicit color distributions can be more convenient than palettes of single-color values. Aksoy et al. [7] constructed a small number of representative color distribution models for a linear blending model using a Gaussian kernel in RGB space. In contrast, our framework estimates the missing colors of an input image that cannot be represented by the given palette.

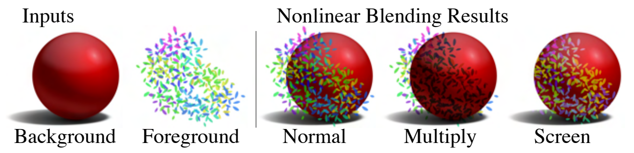


Figure 2. Examples of visual effects obtained by using different blend modes. The *normal* blend mode simply selects the foreground color, whereas the *multiply* and *screen* blend modes produce a darkening and a lighting effect, respectively.

2.2. Soft Color Segmentation

Soft color segmentation is the problem of decomposing an image into multiple semi-transparent layers, each of which has nearly homogeneous colors. Soft color layer representation is crucial for color editing because it allows the layers to be edited individually. Aksoy et al. [7] proposed a color unblending formulation for a linear blending model. FSCS proposed by Akimoto et al. [5] proposed a first effective convolutional neural network (CNN)-based soft color segmentation method, which they demonstrated to be 300,000 times faster than an optimization-based method [7]; however, this method was only used for a linear blending model.

DIL proposed by Koyama et al. [15] performed a first soft color segmentation with advanced blending using an optimization method, particularly an augmented Lagrangian method. However, their method requires iterative optimization and is highly time consuming for each inference. To address this drawback, we propose a neural-network-based soft color segmentation method that can handle advanced blending and achieve faster inference.

3. Proposed Approach

3.1. Overall Framework

Our goal in this approach is to train models to decompose a given image into a set of semi-transparent layers un-

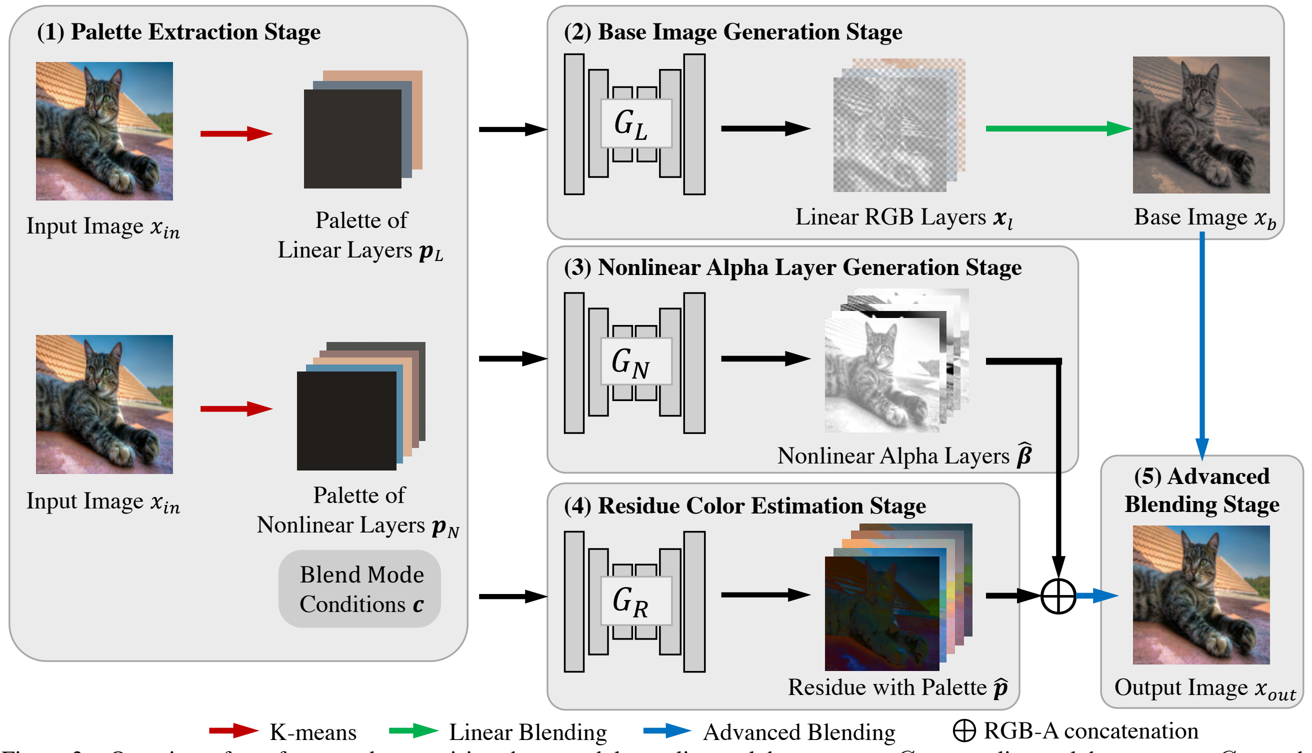

(1) Palette Extraction Stage
Input Image x_{in}
Palette of Linear Layers p_L
Input Image x_{in}
Palette of Nonlinear Layers p_N
Blend Mode Conditions c
(2) Base Image Generation Stage
G_L
Linear RGB Layers x_l
Base Image x_b
(3) Nonlinear Alpha Layer Generation Stage
G_N
Nonlinear Alpha Layers $\hat{\beta}$
(4) Residue Color Estimation Stage
G_R
Residue with Palette $\hat{p}$
(5) Advanced Blending Stage
Output Image x_{out}
K-means
Linear Blending
Advanced Blending
⊕ RGB-A concatenation

Figure 3. Overview of our framework comprising three modules: a linear alpha generator G_L, a nonlinear alpha generator G_N, and a residue generator G_R. Image courtesy of Gabriel González.

der the constraint of the blend modes $\{c_1, c_2, ..., c_n\}$ and palette RGB colors $\{v_1, v_2, ..., v_n\}$, where n denotes the target number of nonlinear layers. Our framework consists of three generative networks: a linear alpha generator G_L, a nonlinear alpha generator G_N, and a residue generator G_R, as shown in Figure 3. Given the color palettes, our framework predicts a linear alpha layer $\boldsymbol{\alpha} \in \mathbb{R}^{m \times H \times W \times 1}$ and a nonlinear alpha layer $\boldsymbol{\beta} \in \mathbb{R}^{n \times H \times W \times 1}$, where m and n denote the numbers of linear and nonlinear alpha layers, respectively. H and W represent the height and width of the input image, respectively. To promote the use of the nonlinear layers, we intentionally restrict the number of linear alpha layers to a small number (e.g., three).

Our proposed framework consists of five stages: (1) In the *palette extraction stage*, a set of palette colors is determined by either an automatic algorithm or manual selection. Each selected palette color indicates the mean color of a soft color layer. (2) In the *base image generation stage*, the linear alpha generator G_L decomposes the input image into linear alpha layers based on the palette colors. Our framework determines the base image by linearly blending the linear alpha layers. (3) In the *nonlinear alpha layer generation stage*, the nonlinear alpha generator G_N predicts the nonlinear alpha layers given the blend modes and palette colors. (4) In the *residue color estimation stage*, the residue generator G_R estimates the colors that are not included in the palette colors. (5) Finally, in the *advanced blending stage*, the outputs of stages (2)–(4) are composited to reconstruct the original input image. The nonlinear alpha layers in stage (3) and residue colors in stage (4) together constitute the nonlinear soft color layers, which are then applied to the base image using advanced blending.

3.2. Palette Selection

When creators use the soft color segmentation method to decompose an image, they should be able to select colors that represent the colors of the soft color layers. For training our framework, we simulate such behavior by automatically determining the palette colors of the soft color layers. To be specific, we use K-means, as in FSCS [5], as an automatic palette extraction method to cluster the pixels in the input image x_{in} in RGB space. We define the i-th palette RGB value $v_i \in \mathbb{R}^3$ as the centroid of the i-th cluster. The palette color of each layer $p_i \in \mathbb{R}^{H \times W \times 3}$ is an RGB layer of color v_i on each pixel. K-means extracts the color palettes of the linear layers $\boldsymbol{p}_L \in \mathbb{R}^{m \times H \times W \times 3}$ and the palette of the nonlinear layers $\boldsymbol{p}_N \in \mathbb{R}^{n \times H \times W \times 3}$. In this way, K-means captures the main colors of an image, and therefore also serves as a convenient starting point for a manual selection in practice.

3.3. Linear Alpha Generator

In contrast with Koyama's method [15], which uses a base image as the first layer, our approach linearly blends the linear soft color layer to construct the base image as a rough approximation of the input image. Specifically, given the input image x_{in} and the palette of linear layers $\boldsymbol{p}_L$, the linear alpha generator G_L predicts the linear alpha layers $\boldsymbol{\alpha}$, which is then normalized to obtain the preprocessed alpha layers $\hat{\boldsymbol{\alpha}}$ formulated as

$$\hat{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i \oslash \sum_{k=1}^{m} \boldsymbol{\alpha}_k, \;\; \boldsymbol{\alpha} = G_L(x_{in}, \boldsymbol{p}_L), \tag{1}$$

where $\oslash$ denotes the Hadamard division, namely the per-element matrix division. Here, $\hat{\boldsymbol{\alpha}}_i$ denotes the i-th normalized alpha layer, and m denotes the number of linear alpha layers. The linear RGB layers $\hat{\boldsymbol{\alpha}}$ are then defined to have the RGB values of palette $\boldsymbol{p}_L$ weighted by alpha values of $\hat{\boldsymbol{\alpha}}$. Linear blending of the linear RGB layers $\boldsymbol{x}_l$ yields the base image x_b. The base image later serves as the background in the advanced blending stage.

3.4. Nonlinear Alpha Generator

The nonlinear alpha generator G_N predicts the nonlinear alpha layers $\boldsymbol{\beta}$. Because changing the blend modes or palette colors will produce different optimal alpha layers, we explicitly enforce such control of the blend modes and palette colors over the generated alpha layers in the proposed framework. The adaptive instance normalization (AdaIN) [11] has been adopted by a number of image-to-image translation works [8, 12, 18, 23] and takes a style code, which is usually encoded from a reference image, as a parameter to manipulate the output style. We employ this module in the architecture of the nonlinear alpha generator. The blend mode and palette color details are embedded into a code that parameterizes the AdaIN module, which in turn modulates the generation of the nonlinear alpha layers. As shown in Figure 4, a mapping network embeds the RGB palette colors of the nonlinear layers $\boldsymbol{v}$ and the blend mode condition $\boldsymbol{c}$ into the code $w \in \mathbb{R}^{64}$. Thanks to AdaIN, G_N is capable of generating nonlinear alpha layers based on specified blend modes and palette colors. Then, β are normalized by Eq. 1 and $\hat{\boldsymbol{\beta}}$ are produced.

3.5. Residue Generator

The input palettes calculated by the K-means method preserve only partial color information of the input image. To recover the full color information of and reconstruct the input image, our framework predicts the remaining colors. Specifically, G_R predicts the residue colors $\boldsymbol{r}$ of the palette of nonlinear layers $\boldsymbol{p}_N$. Similar to the nonlinear alpha generator G_N, G_R also contains the AdaIN modules in its normalization layers. This step is formulated as

$$\begin{aligned} \hat{\boldsymbol{p}}_N &= \boldsymbol{p}_N + \tanh\left(\boldsymbol{r} - \frac{\sum_{i=1}^{HW} \boldsymbol{r}_i}{HW}\right), \\ \boldsymbol{r} &= G_R(x_{in}, \boldsymbol{p}_N, \boldsymbol{c}, \boldsymbol{v}), \end{aligned} \tag{2}$$

where $\boldsymbol{r}_i$ denotes the residue value for the i-th pixel. $\hat{\boldsymbol{p}}_N$ represents the RGB layers of the color palettes with residues. To calculate zero-centered $\hat{\boldsymbol{p}}_N$, the generated residues $\boldsymbol{r}$ are subtracted from the spatial means of their RGB values. We finally obtain the nonlinear RGBA layers $\boldsymbol{x}_r$ by taking the RGB values from $\hat{\boldsymbol{p}}_N$ and alpha values from the nonlinear alpha layers $\hat{\boldsymbol{\beta}}$. We denote $\boldsymbol{x}_r$ using boldface since it is a stack of layers.

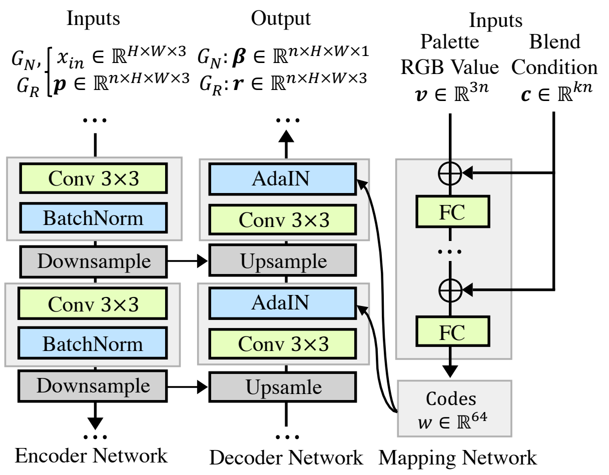

Inputs
G_N, G_R $x_{in} \in \mathbb{R}^{H\times W\times 3}$ $\boldsymbol{p} \in \mathbb{R}^{n\times H\times W\times 3}$
Output
G_N: $\boldsymbol{\beta} \in \mathbb{R}^{n\times H\times W\times 1}$
G_R: $\boldsymbol{r} \in \mathbb{R}^{n\times H\times W\times 3}$
Inputs
Palette RGB Value $\boldsymbol{v} \in \mathbb{R}^{3n}$
Blend Condition $\boldsymbol{c} \in \mathbb{R}^{kn}$
Conv 3×3
BatchNorm
Downsample
AdaIN
Upsample
Upsamle
FC
Codes $w \in \mathbb{R}^{64}$
Encoder Network
Decoder Network
Mapping Network

Figure 4. Generator architecture of both the nonlinear alpha generator G_N and the residue generator G_R. Here, w denotes the scaling and shifting features of the AdaIN method.

The advantages of employing the residue estimation include not only quantitative improvement, as shown in Table 3, but also natural implementation of palette editing. As $\hat{\boldsymbol{p}}_N$ can be regarded as a distribution of colors centered at $\boldsymbol{p}_N$, we can modify the center value to shift the distribution. This naturally introduces the recoloring operation of an image. For example, we can modify the palette of the leftmost image in Figure 1 to obtain the rightmost recolored image.

3.6. Advanced Blending

Advanced blending is the process of composition that mixes colors by overlapping the source (foreground) and destination (background). This approach comprises two processes, namely *blending* and Porter–Duff compositing [21]. *Blending* determines how the foreground and background RGB values are composited. Given the RGBA values of the foreground and background layers, the RGB values of the blended layer are calculated from the foreground and background RGB values using the blending function of a specified blend mode. The alpha value of the blended layer is simply set to that of the foreground layer. Then, Porter–Duff compositing is used to calculate the final RGBA value by considering the contributions of the blended and background RGBA values for each pixel. Here, we use the *source-over* operator as the Porter–Duff compositing operator because advanced blending only adopts the source-over operator in practice, and the major authoring tools [1, 2] only support the *source-over* operator. For more details, please see the official definition [4].

In the advanced blending stage of our framework, we use the base image x_b as the background image and the nonlinear RGBA layers $\boldsymbol{x}_r$ to composite the output image x_{out}. Since there are multiple nonlinear RGBA layers, we iteratively perform advanced blending: we start by using the base image as the background image, and for each step, advanced blending takes a nonlinear layer as the source image and the last output of the iteration as the source image.

3.7. Objectives

Our framework consists of three generator networks with the following training objectives.

Reconstruction loss. We define the mean absolute error between the input and composite images as follows:

$$\mathcal{L}_b = \|x_{in} - x_b\|_1, \tag{3}$$

$$\mathcal{L}_o = \|x_{in} - x_{out}\|_1, \tag{4}$$

where the base image x_b and advanced blended image x_{out} are the outputs of stages (2) and (5), respectively.

SCU loss is inspired by the sparse color unmixing (SCU) energy objective proposed by Aksoy et al. [6, 7]. We introduce this objective to prevent large deviations of the residues from the palette colors while promoting the use of alpha values, both of which enable the generation of reasonably advanced blending layers. The SCU loss $\mathcal{L}_{scu}$ is defined as

$$\mathcal{L}_{scu} = \sum_{i=1}^{n} \hat{\boldsymbol{\beta}}_i \|\boldsymbol{p}_{N_i} - \hat{\boldsymbol{p}}_{N_i}\|_2, \tag{5}$$

where the normalized nonlinear alpha layers$\hat{\boldsymbol{\beta}}$ and palettes with residues $\hat{\boldsymbol{p}}$ are the outputs of stage (3) and (5), respectively. i denotes the layer number of each nonlinear palette and alpha layer.

Total loss is optimized in our framework. It is defined as

$$\mathcal{L}_{total} = \lambda_b \mathcal{L}_b + \lambda_o \mathcal{L}_o + \lambda_{scu} \mathcal{L}_{scu}, \tag{6}$$

where λ_b, λ_o, and λ_{scu} are hyperparameters described in Section 4.1.

4. Experiments

4.1. Implementation Details

Dataset. We used the Places365-Standard [28] validation set for our training and evaluation. The dataset contains 36,500 images of scenes such as "amusement park" and "street." We divided the images into training and test sets containing 35,500 and 1,000 images, respectively. We then resized the images to a resolution of 256×256 and normalized the color values in the range of $[0, 1]$.

Training. Our framework was trained using Adam [14] with $\beta_1 = 0.0$ and $\beta_2 = 0.99$, and initialized using He initialization [10]. We set the learning rate for our networks to $2e^{-4}$. Then, we randomly sampled n blend modes from among the k blend modes as a condition vector in each iteration. The batch size was set to 16, and our framework was trained for 200 epochs. The training required approximately one day with four V100 GPUs in PyTorch [19]. For the hyperparameters, we set $\lambda_b = \lambda_o = 20$ and $\lambda_{scu} = 5$. First of a 10 epoch, we trained the linear alpha generator using Eq (3) to learn reconstruction of a base image. Then, we trained three generators using Eq (6). We describe the details of our proposed network architecture in the supplementary material.

Table 1. **Quantitative comparison** with the baseline methods. NL indicates that the method is for nonlinear blending.

Methods	NL	Image Quality			
		SSIM↑	PSNR↑	MSE↓	LPIPS↓
RGBXY [24]		0.997	45.82	0.000038	0.0023
FSCS [5]		0.972	32.41	0.00063	0.0165
DIL [15]	✓	0.991	43.94	0.00058	0.0068
Ours	✓	0.983	35.09	0.00031	0.0117

4.2. Baseline Methods

The baseline methods used in our comparisons are RGBXY [24], DIL [15], and FSCS [5].

RGBXY [24] is a method to decompose an image into a set of linear blending layers It consists of two steps: (1) computation of the RGBXY convex hull (instead of using given palettes) and (2) update of the layer.

DIL [15] is a method to decompose an image into a set of nonlinear layers using the optimization method. The DIL method requires not only color palettes but also their corresponding color distribution parameters of each layer.

FSCS [5] is a method to perform first soft color segmentation with a CNN. It also predicts the residues to identify colors that are not included in a given palette. The FSCS method is limited to linear blending models.

4.3. Quantitative Evaluation

Quantitative scores. As shown in Table 1, we compared the different methods in terms of the structural similarity index (SSIM), peak signal-to-noise ratio (PSNR), mean-squared error (MSE), and learned perceptual image patch similarity (LPIPS) [27] for the test set with a resolution of 256×256. To calculate the LPIPS, we used the ImageNet-pretrained AlexNet [16] as the feature extractor. RGBXY determined the palette size through an optimization and used the number of an average and median palette size 6.95 and 7, respectively. For the other methods explored here, we set the palette size to 7. For our proposed method and DIL [15], we used a random blend mode from among the 12 advanced blend modes for each nonlinear layer. The color variance parameters of the DIL were set to 0.3.

The RGBXY [24] method achieved the best score on the image quality metrics, indicating that the iterative method

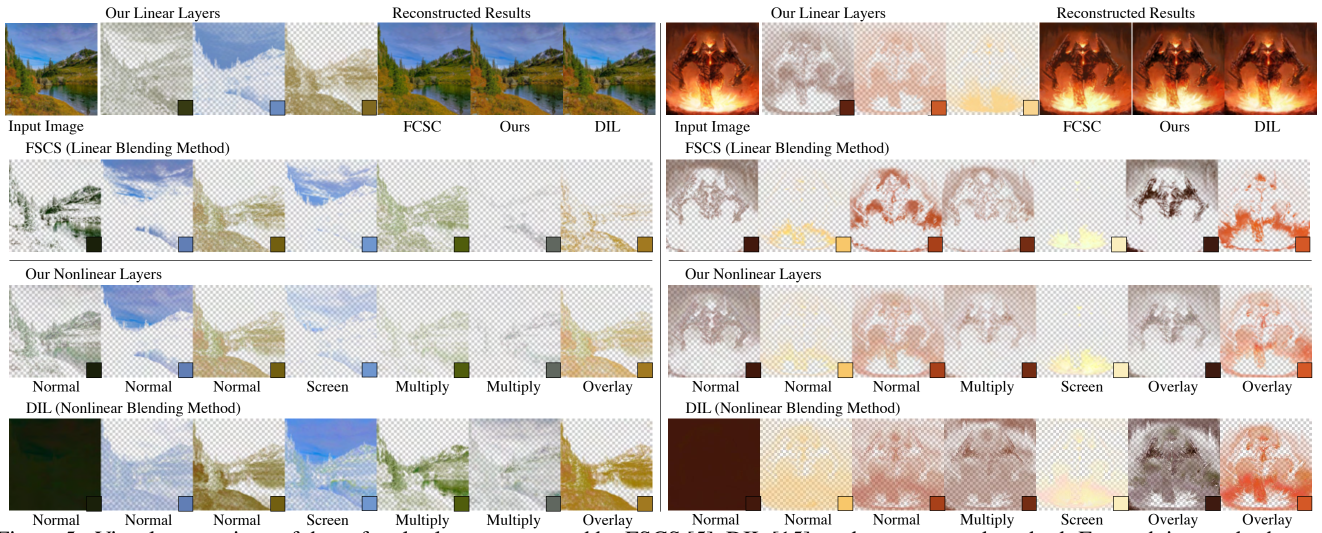


Figure 5. Visual comparison of the soft color layers generated by FSCS [5], DIL [15], and our proposed method. For each image, both our method and DIL use the same blend modes for nonlinear layer decomposition. The same palette is used for FSCS, DIL, and the nonlinear alpha layer generation stage of our method. The right image is courtesy of David Revoy.

of the linear blending model performs well. Additionally, the DIL [15] achieved higher scores than the proposed and FSCS [5] neural-network-based methods. Compared with the FSCS [5], which is a linear blending model, our proposed method achieved better scores.

Inference time. Table 2 provides a comparison of the inference times needed by the baseline and proposed methods to decompose a given image using the seven palettes. We performed the decomposition using an Intel Core i7-9700K 3.6 GHz CPU and an NVIDIA V100 GPU and calculated the average values of 20 repetitions. The inference times of FSCS, DIL, and ours were increasing linearly. The inference time of RGBXY [24], which is a linear blending model, was approximately 70–140 times greater than that of our method with a GPU. The FSCS [5] and proposed methods are sufficiently fast for digital creators to perform quick trial-and-error procedures. The DIL [15] approach, which performs nonlinear layer decomposition, required the greatest inference time; a 2-megapixel image required 72 s for decomposition, making it difficult to perform trial-and-error procedures to determine the desired blend modes and palette color. In comparison, our method achieved the same decomposition in 0.2 s with a GPU, which is approximately 370 times faster.

Table 2. **Inference time comparison**. MP and NL refer to megapixels and nonlinear blending, respectively.

Methods	NL	Inference Time			
		2MP[s]	4MP[s]	6MP[s]	8MP[s]
RGBXY		27.5	33.8	50.3	55.6
FSCS (CPU)		3.16	6.32	9.55	13.0
FSCS (GPU)		0.07	0.14	0.21	0.28
DIL	✓	72.4	145.5	216.8	291.6
Ours (CPU)	✓	6.47	12.4	17.5	24.6
Ours (GPU)	✓	0.19	0.38	0.58	0.77

4.4. Qualitative Evaluation

Figure 5 shows the results of the linear and nonlinear decomposition and the reconstructed image. For each image, both our method and DIL use the same blend modes for nonlinear layer decomposition. The same palette is used for FSCS, DIL, and the nonlinear alpha layer generation stage of our method.

Linear method. We compared linear layers decomposed by our method with those decomposed by FSCS [5], as shown in the top two rows of Figure 5. FSCS used an SCU energy objective in addition to a reconstruction loss to train the alpha and the residue predictors. On the other hand, we trained our linear alpha generator with only a reconstruction loss Eq. (3). As a result, while FSCS is ideal for linear image unblending, our method generates linear layers that composite a minimal base image for nonlinear unblending.

Nonlinear method. We compared nonlinear layers decomposed by our method with those decomposed by DIL [15], as shown in the bottom two rows of Figure 5. DIL tended to produce nonlinear layers with large alpha values (i.e. high opacity), regardless of the blend modes. Additionally, as shown in the second and fourth layers in the left-hand side of the bottom row in Figure 5, DIL produced unrelated regions (e.g. mountain) even though blue palette color was given, while our method generated the soft layer in the region (e.g. sky) corresponding to the blue palette.

Table 3. **Ablation study**. Ablation experiments for the hyperparameters and architectures. All experiments used three palettes for the linear layers. m and k denote the number of palettes for the nonlinear and blend modes. Variance refers to the residue color variance. (d)$\sim$(f) networks are trained with *m*=7 and *k*=4.

Methods	SSIM↑	PSNR↑	MSE↓	LPIPS↓	Variance↓
(a) *m*=7, *k*=4	0.985	35.82	0.00029	0.0129	0.0032
(b) *m*=7, *k*=12	0.979	35.09	0.00041	0.0147	0.0034
(c) *m*=5, *k*=4	0.980	34.32	0.00037	0.0151	0.0027
(d) w/o $\mathcal{L}_{scu}$	0.981	35.01	0.00026	0.0107	0.029
(e) w/o Zero-Centered	0.981	34.34	0.00039	0.0164	0.067
(f) w/o Residue	0.957	33.75	0.00120	0.0275	-

This is a side effect of the optimization objective of DIL that forces nonlinear layers to reconstruct the input image without semantically meanings. On the other hand, while our method behaves similarly when the blend mode is *normal* the decomposed nonlinear layers have small alpha values (i.e. rather transparent) when the blend mode is *overlay*, *screen*, or *multiply*. Thanks to the base image that captures the most basic colors of the input image, nonlinear layers do not struggle to reconstruct, but instead focus on creating the appropriate effect of the advanced blend mode.

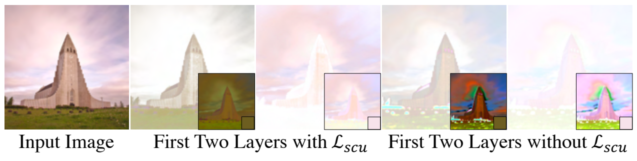
4.5. Ablation Study

Number of blend modes. As shown in Table 3, the scores for case (a) were better than those for case (b). Our method, which uses fewer blend modes, achieved better scores because it was easier to train the networks with fewer blend modes and create the reconstructed result. For seven palettes and 12 blend modes (*m*=7 and *k*=12), a total of $7^{12} = 13,841,287,201$ possible blend mode combinations are obtained, and it is difficult for the network to generalize to all possible cases during training.

Number of palettes of the nonlinear layers. To investigate the effect of the number of layers on advanced blending, we compared case (a) with seven (m=7) and case (c) with five palettes (m=5). In particular, the result of case (a) was superior to that of case (c) for all metrics except for the variance score. We see that the network generated higher quality and more sufficient reconstructions with a larger number of palettes.

SCU loss. The MSE and LPIPS scores in case (d) were superior to those of case (a). This is because, in case (d), there were no restrictions on the color variations of the residue palette, making it easier to reconstruct. Therefore, as shown in Figure 6, case (d) produced residue colors that is not limited to given palette colors and had a higher color variance score. If a residue variance is large, it prevents users from editing the palette. This suggested that $\mathcal{L}_{scu}$ is necessary for adequate learning of the color distribution of the residue.

Residue generator. We compared cases (a) and (e) to evaluate the residue generator. The palette of nonlinear layers with residues in case (a) had a color distribution centered at the original palette color, whereas the one in case (e) directly predicted the distribution with a range of values from $[0, 1]$. Case (a) was better than case (e) for all scores, indicating that the residue generator with zero-centered output is more effective. Finally, compared with case (f), which did not include residues, the results had worse scores than those of case (a). We thus conclude that the residue generator is essential to the quality of our framework.


Input Image
First Two Layers with $\mathcal{L}_{scu}$
First Two Layers without $\mathcal{L}_{scu}$
Figure 6. Comparison of generated nonlinear layers with and without $\mathcal{L}_{scu}$. The palette with residue and palette color are shown in the square on the bottom right of each image.

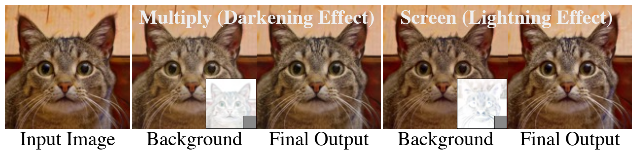


Figure 7. Examples of decomposition with *multiply* and *screen* as the final blend mode. The background, which are the last interim results of reconstruction, and the alpha layers are in the square on the bottom right of each image. Thanks to the visual effect of a blend mode, the final output image successfully reconstructs the input image.

4.6. Discussion

Output control by AdaIN.

We demonstrate that AdaIN controls the nonlinear layer according to the input blend modes and palette colors in Figure 7. We blended the background and corresponding alpha layers to generate final outputs. Both background images are the last interim result of reconstruction. Both alpha layers have the same gray palette color. The output images reflect the visual effects of edge darkening with *multiply* and eye highlighting with *screen* thanks to appropriate alpha layers and residue colors with AdaIN conditioning. Therefore, thanks to AdaIN, the nonlinear alpha generator is capable of controlling the generation of nonlinear layers depending on the blend modes.

Number of linear layers. Figure 8 shows a comparison of the base images using palette sizes of three and seven. The base image forms the first layer for advanced blending. Hence, having more palette colors for linear layers makes a color editing more complicated. In this work, the number of palettes for linear layers is set to three.

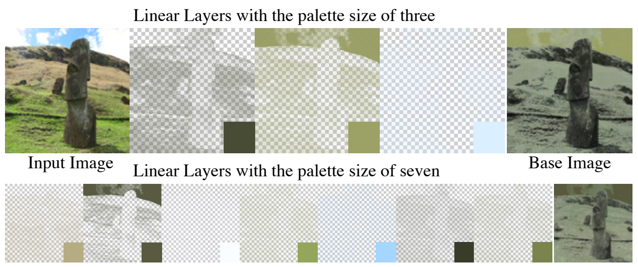

Linear Layers with the palette size of three
Input Image
Base Image
Linear Layers with the palette size of seven

Figure 8. Comparison of the base images using palette sizes of three and seven. The palette color is shown in the square on the bottom right of each image. The rightmost image shows the result of reconstruction by linear blending.

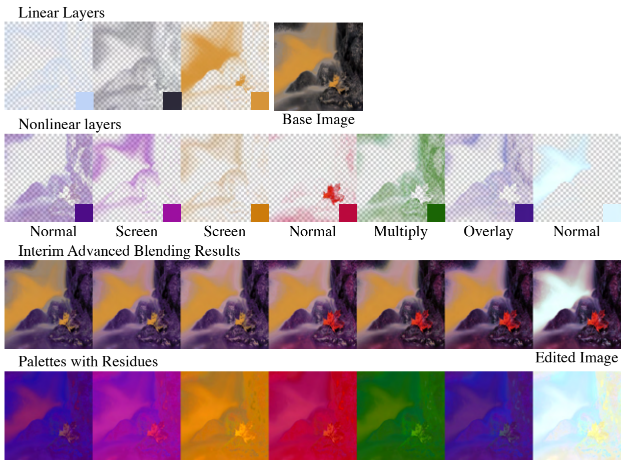


Figure 9. Palette color editing process of the edited image in Figure 1. In the blending step, we use new color palettes for linear and nonlinear layers, while alpha layers remain unchanged. Image courtesy of Theo Crazzolara.

4.7. Applications to Image Editing

Our method can be applied to image editing that requires advanced blending. Our method first decomposes an RGB image into soft color layers—linear and nonlinear layers—using the automatically determined blend modes and palettes. A user can then adjust the soft color layers by manually altering the blend modes and palettes. After satisfied with the soft color layers, the user can start editing the image by, e.g., shifting palette colors or switching blend modes. He or she can come back anytime to exploration of more blend modes and palettes to attain an ideal set of soft color layers for editing. Every time the blend mode or palette changed, our method infers the soft color layers in a fraction of second. Consequently, such a trial-and-error process incurs little waiting time.

Some applications to image editing include:

Editing palette color. Figures 1 and 9 demonstrate results for editing palette color. We keep blend modes the same during unblending and blending, while changing to another set of palette colors for soft color layers.

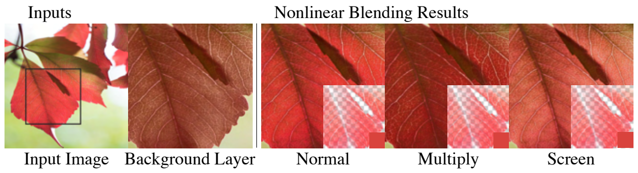


Figure 10. Examples of blend mode editing to create different visual effects. For decomposition, we specify *normal* as the blend mode of one of the nonlinear layers. For composition, we set the blend mode of the nonlinear layer to either *normal*, *multiply*, or *screen*. Image courtesy of Susanne Nilsson.

Editing blend mode. Figure 10 shows results for editing blend modes. Keeping the blend mode as *normal* results in the reconstruction of the input image. Switching the blend mode to either *multiply* or *screen*, on the other hand, creates either a darker or brighter version.

4.8. Limitations

Fixed number of palettes and blend modes. Once we train our networks with a fixed number of palettes and blend modes, the network cannot be used with a different number of palettes and blend modes. For example, the generators for the nonlinear layers trained with five palettes or four blend modes cannot handle seven palettes or twenty blend modes. Adaptation to the different numbers of palettes or blend modes thus remains a problem.

GPU memory limitation. For fast inference, we accelerate our method using GPUs. In this case, the size of the input image is restricted by the memory of the GPU (e.g., for a GPU with 12 GB memory, the input images cannot be larger than 4 MP). Using the half-precision (e.g., FP16) format instead of the single-precision (FP32) format may thus mitigate this issue.

5. Conclusion

We proposed the first neural-network-based method for nonlinear decomposition according to the specified color palettes and blend modes. We have experimentally verified that our method achieved an inference speed 370 times faster than the state-of-the-art method of nonlinear image unblending. Furthermore, qualitative and quantitative comparisons have demonstrated that our reconstruction quality was higher than or comparable to those of other methods, including linear blending models. To our best knowledge, our method opens the door to fast editing of real-world photographs using blend modes.

6. Acknowledgements

We would like to thank Yanghua Jin and Edgar Simo-Serra for their insightful feedback on our project. Daichi Horita was supported by Toyota and Dwango Scholarship. This work was partially supported by JSPS KAKENHI Grant Number 21H03460.

References

[1] ADOBE SYSTEMS INC.: Adobe Photoshop CC. `https://www.adobe.com/products/photoshop.html`.

[2] KDE: Krita. `https://krita.org/`.

[3] W3C: SVG compositing specification. `https://www.w3.org/TR/2011/WD-SVGCompositing-20110315/`, 2011.

[4] W3C: Compositing and blending level 1. `https://www.w3.org/TR/2015/CR-compositing-1-20150113`, 2015.

[5] Naofumi Akimoto, Huachun Zhu, Yanghua Jin, and Yoshimitsu Aoki. Fast Soft Color Segmentation. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Yağız Aksoy, Tunç Ozan Aydın, Marc Pollefeys, and Aljoša Smolić. Interactive High-Quality Green-Screen Keying via Color Unmixing. *ACM Trans. Graph.*, 35(5):152:1–152:12, 2016.

[7] Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Trans. Graph.*, 36(4), Mar. 2017.

[8] Ivan Anokhin, Pavel Solovev, Denis Korzhenkov, Alexey Kharlamov, Taras Khakhulin, Alexey Silvestrov, Sergey Nikolenko, Victor Lempitsky, and Gleb Sterkin. High-Resolution Daytime Translation Without Domain Labels. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2020.

[9] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based Photo Recoloring. *ACM Trans. Graph.*, 34(4), 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015.

[11] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[12] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal Unsupervised Image-to-Image Translation. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[13] Taehong Jeong, Myunghyun Yang, and Hyun Joon Shin. Succinct Palette and Color Model Generation and Manipulation using Hierarchical Representation. *Computer Graphics Forum*, 38(7):1–10, 2019.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. the International Conference on Learning Representation (ICLR)*, 2015.

[15] Yuki Koyama and Masataka Goto. Decomposing Images into Layers with Advanced Color Blending. *Computer Graphics Forum*, 37, 2018.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2012.

[17] Sharon Lin, Matthew Fisher, Angela Dai, and Pat Hanrahan. LayerBuilder: Layer Decomposition for Interactive Image and Video Color Editing. arXiv preprint arxiv/1701.03754, 2017.

[18] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-Shot Unsupervised Image-to-Image Translation. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2019.

[20] Huy Q. Phan, Hongbo Fu, and Antoni B. Chan. Color Orchestra: Ordering Color Palettes for Interpolation and Prediction. *IEEE Transactions on Visualization and Computer Graphics*, 24(6):1942–1955, 2018.

[21] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, page 253–259, 1984.

[22] Zhang Qing, Chunxia Xiao, Hanqiu Sun, and Feng Tang. Palette-Based Image Recoloring Using Color Decomposition Optimization. *IEEE Transactions on Image Processing*, 26(4):1952–1964, 2017.

[23] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. COCO-FUNIT: Few-Shot Unsupervised Image Translation with a Content Conditioned Style Encoder. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[24] Jianchao Tan, Jose Echevarria, and Yotam Gingold. Efficient Palette-Based Decomposition and Recoloring of Images via RGBXY-Space Geometry. *ACM Trans. Graph.*, 37(6), 2018.

[25] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing Images into Layers via RGB-Space Geometry. *ACM Trans. Graph.*, 36(1), Nov. 2016.

[26] Lvmin Zhang, Edgar Simo-Serra, Yi Ji, and Chunping Liu. Generating Digital Painting Lighting Effects via RGB-space Geometry. *ACM Trans. Graph.*, 39(2), 2020.

[27] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.