# Unpaired Image Enhancement Featuring Reinforcement-Learning-Controlled Image Editing Software

**Satoshi Kosugi, Toshihiko Yamasaki**
Department of Information and Communication Engineering, The University of Tokyo, Tokyo, Japan
{kosugi, yamasaki}@hal.t.u-tokyo.ac.jp

**Abstract**

This paper tackles unpaired image enhancement, a task of learning a mapping function which transforms input images into enhanced images in the absence of input-output image pairs. Our method is based on generative adversarial networks (GANs), but instead of simply generating images with a neural network, we enhance images utilizing image editing software such as Adobe® Photoshop® for the following three benefits: enhanced images have no artifacts, the same enhancement can be applied to larger images, and the enhancement is interpretable. To incorporate image editing software into a GAN, we propose a reinforcement learning framework where the generator works as the agent that selects the software's parameters and is rewarded when it fools the discriminator. Our framework can use high-quality non-differentiable filters present in image editing software, which enables image enhancement with high performance. We apply the proposed method to two unpaired image enhancement tasks: photo enhancement and face beautification. Our experimental results demonstrate that the proposed method achieves better performance, compared to the performances of the state-of-the-art methods based on unpaired learning.

## Introduction

Image enhancement is a task of learning a mapping function which transforms input images into enhanced images. If we have a large number of original and enhanced image pairs, the task can be solved by image-to-image translation methods, which have made significant progress (Isola et al. 2017; Wang et al. 2018) owing to the recent development of convolutional neural networks (CNNs). However, in many cases, it is difficult to collect a large number of such image pairs. To avoid this problem, we address an image enhancement task that does not require paired image datasets; that is, unpaired image enhancement. In this paper, we propose an unpaired image enhancement method which can be applied to real-world tasks.

A simple approach for unpaired image enhancement can be to use unpaired image-to-image translation methods, which are mainly based on generative adversarial networks (GANs) (Goodfellow et al. 2014). One of such methods is CycleGAN (Zhu et al. 2017), where generators which have an encoder-decoder architecture are trained with cycle consistency. However, when using CNNs as decoders in real-world tasks, there are three problems. First, images generated by a CNN-based decoder have artifacts that can be attributed to CNN architecture. Because artifacts can seriously degrade the quality of images, they can have fatal defects when used in practical applications. Second, CNN-based decoders can only generate images with limited resolution in practice (*e.g.*, 512×512px in the CycleGAN paper). Recent high-resolution displays need 2000px or larger images, but generating images with a high resolution makes the training unstable and time-consuming. Third, image-to-image translation with CNN-based decoders is not interpretable. Because the procedure is black-box, users cannot understand and manually adjust it.

To achieve unpaired image enhancement that is without artifacts, is scale-invariant, and is interpretable, we use image editing software which edits the input image based on input parameters, such as Adobe Photoshop. Using image editing software in the processing flow has the following three benefits: edited images have no artifacts because the software is carefully designed for professional use, the same editing can be applied to large sized images using the scale-invariant image editing filters provided by the software, and the editing is interpretable allowing users to easily adjust it manually. By using image editing software, we can achieve high-quality and highly practical image enhancement. To utilize image editing software in a GAN, we propose a reinforcement learning (RL) framework where the generator works as the agent controlling the software. While a generator in a general GAN generates images directly, the generator in our framework selects the software's parameters and is rewarded when the edited result fools the discriminator. By training the framework with RL, we can use high-quality non-differentiable image editing software.

To evaluate the performance of the proposed method, we apply it to two unpaired image enhancement tasks: photo enhancement and face beautification. The experimental results show that the proposed method achieves better performance than previous approaches.

This paper makes the following contributions:

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

- We achieve unpaired image enhancement that is without artifacts, is scale-invariant, and is also interpretable.
- We use image editing software and propose an RL framework to incorporate image editing software into a GAN. The generator is trained as the agent to select the software's parameters and is rewarded when it fools the discriminator.
- We apply the proposed framework to the tasks of photo enhancement and face beautification.

## Related Works

### Image-to-Image Translation

We formulate image enhancement as a task of learning the mapping from original images to images with the desired characteristics, which is one of image-to-image translation problems. A major CNN-based method for image-to-image translation is pix2pix (Isola et al. 2017), which uses a conditional GAN (Goodfellow et al. 2014) to learn a mapping from source to target images. Based on this method, Wang et al. (2018) achieved image-to-image translation with high resolution using multi-scale generators and discriminators. These paired-learning methods require a large number of pairs of input and output images, but in many cases, such pairs of images cannot be obtained. To solve this problem, Zhu et al. (2017) developed an unpaired image-to-image translation technique named CycleGAN, where two GANs are trained using cycle consistency. Kim et al. (2017) and Yi et al. (2017) also proposed similar methods and named them DiscoGAN and DualGAN, respectively. Choi et al. (2018) proposed a method named StarGAN that can handle translation between multiple domains. We propose a more practical method than applying these methods directly to image enhancement.

### Reinforcement Learning for Image Processing

In recent years, deep RL is being applied to image processing. Cao et al. (2017) applied RL to super-resolution of facial images. In that study, areas to be enhanced are sequentially selected by RL. Li et al. (2018) proposed an RL-based image cropping method, where an agent sequentially updates the cropping window enabling high-speed cropping. Yu et al. (2018) used RL to select a toolchain from a toolbox for image restoration. Furuta et al. (2019) proposed a fully convolutional network that allows agents to perform pixel-wise manipulations.

One of the benefits of RL is that a framework containing non-differentiable functions can be optimized. Ganin et al. (2018) proposed a reinforced adversarial learning method for synthesizing simple images of letters or digits using a non-differentiable renderer. Because the image editing software we use and its renderer are both non-differentiable, we apply some of their training strategy to our unpaired image enhancement method.

### Photo Enhancement

Photo enhancement can be formulated as a translation between low-quality original images and high-quality expert-retouched images. Bychkovsky et al. (2011) created a large-scale paired dataset for photo enhancement. They hired five expert retouchers and created a collection of five sets of 5,000 input-output image pairs. Using this paired dataset, Yan et al. (2016) proposed an automatic photo adjustment framework, which considers the local semantics of an image. Gharbi et al. (2017) developed a CNN to predict the coefficients of a locally affine model in a bilateral space and achieved high-speed edge-preserving photo enhancement. Wang et al. (2019) built an underexposed image dataset and proposed a network that can handle diverse lighting conditions.

Collecting pairs of original and expert-retouched images is labor-intensive. To address this problem, unpaired learning methods have been proposed. Chen et al. (2018b) made some improvements to CycleGAN to develop a stable two-way GAN framework. Park et al. (2018) created pseudo-input-retouched pairs by randomly distorting high-quality reference images. Hu et al. (2018) proposed a deep RL-based framework that applies retouching operations sequentially. Their method is similar to our proposed method, but their architecture can only use differentiable filters. While the available filters in their framework are limited, our method can use a variety of filters because our method does not require filters to be differentiable. In addition, the same framework can be applied to a completely different task such as face beautification.

### Face Attribute Manipulation

Face beautification, a task of converting a less attractive face into an attractive face, is one application of face attribute manipulation. On of the methods for face attribute manipulation is CycleGAN, but the model is difficult to train, and generated images may include artifacts. Several GAN-based approaches have been proposed to overcome this problem. Shen et al. (2017) achieved efficient face attribute manipulation by generating only the difference between images before and after the manipulation instead of generating the entire image. Zhang et al. (2018) introduced spatial attention to avoid edits in unrelated parts.

Another approach called deep feature interpolation (DFI), which does not use GANs, was proposed by Upchurch et al. (2017). By manipulating the deep features of the input image with a specific attribute vector and performing back-propagation to the image space, the image after the manipulation can be obtained. Using DFI, Chen et al. (2018a) achieved fast and high-quality face attribute manipulation with an end-to-end CNN that learns attribute vectors. Chen et al. (2019) developed a model that decomposes a facial attribute into multiple semantic components, each corresponding to a specific face region. These techniques have produced great results, but face attribute manipulation using CNNs inevitably generates artifacts. This is a serious issue in face beautification.

## Method

Our goal is to learn a mapping function which transforms input images into enhanced images in the absence of input-output image pairs. We formulate this task as unpaired

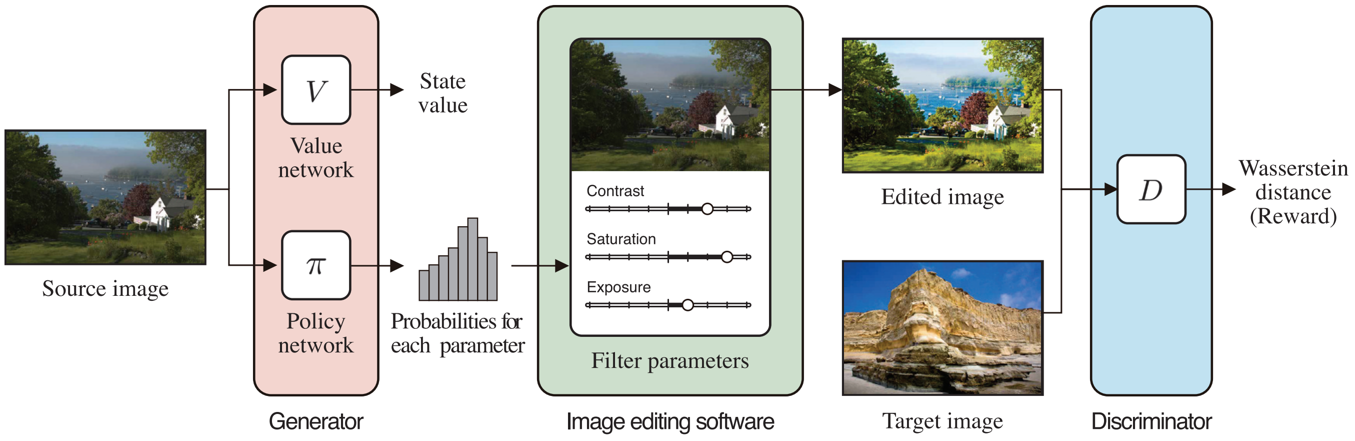


Figure 1: Overview of our method. In our framework, the generator is trained with RL to control image editing software, and the output of the discriminator is used as the reward.

image-to-image translation from source domain $X$ to target domain $Y$, where $X$ and $Y$ contain original images $\{\mathbf{x}_i\}_{i=1}^N$ and images with the desired characteristics $\{\mathbf{y}_j\}_{j=1}^M$, respectively. We denote the data distribution as $\mathbf{x} \sim p_s$ and $\mathbf{y} \sim p_t$. A simple approach can be training a CNN-based generator such as CycleGAN (Zhu et al. 2017). However, CNN-based generators have several problems: the generated image has artifacts, the generator is not scale-invariant, and the translation is not interpretable. To achieve high-quality image enhancement that addresses these problems, we introduce image editing software $\mathcal{S}$ such as Adobe Photoshop. This image editing software $\mathcal{S}$ takes an image $\mathbf{x}$ and an action vector $\mathbf{a} = [a_1, a_2, ..., a_K]$ as input and outputs the edited image $\mathbf{y}' = \mathcal{S}(\mathbf{x}, \mathbf{a})$. Here, $K$ is the number of filters in the image editing software $\mathcal{S}$. To incorporate the image editing software into a GAN, we propose an RL framework, which consists of the image editing software, one generator, and one discriminator. In this framework, the generator works as an agent selecting parameters for the software and is rewarded when it fools the discriminator. Through the training process, the distribution defined by the generator $\mathbf{y}' \sim p_g$ gradually approaches $p_t$. We show the overview of our framework in Figure 1 and give detailed explanations of the discriminator and the generator in the following sections.

## Discriminator

The training process of our discriminator $D$ is the same as that of discriminators in general GANs. That is, it learns to distinguish the generated images from the real images. We follow a method of Wasserstein GAN with gradient penalty (WGAN-GP) (Gulrajani et al. 2017) and define the loss function as follows,

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{y} \sim p_t}[D(\mathbf{y})] + \mathbb{E}_{\mathbf{y}' \sim p_g}[D(\mathbf{y}')] + \lambda Z, \quad (1)$$

where the first and the second terms increase the Wasserstein distance between generated images and real images. $\lambda$ is a weight for $Z$, and $Z$ is a regularization term for the discriminator to stay in the set of Lipschitz continuous functions,

$$Z = \mathbb{E}_{\hat{\mathbf{y}} \sim p_{\hat{\mathbf{y}}}}[(\|\nabla_{\hat{\mathbf{y}}} D(\hat{\mathbf{y}})\|_2 - 1)^2]. \quad (2)$$

$\hat{\mathbf{y}}$ is an image sampled along straight lines between images in $p_t$ and $p_g$.

## Generator

We aim to incorporate image editing software into a GAN framework. That is, our generator takes an original image $\mathbf{x}$ as input and outputs parameters for the software. A simple approach is to design a differentiable image editing software $\overline{\mathcal{S}}$. A generator $\overline{G}$ which generates parameters for $\overline{\mathcal{S}}$ can be directly optimized by minimizing the following loss:

$$\mathcal{L}_{\overline{G}} = -\mathbb{E}_{\mathbf{x} \sim p_s}[D(\overline{\mathcal{S}}(\mathbf{x}, \overline{G}(\mathbf{x})))]. \quad (3)$$

However, this method cannot use non-differentiable software such as Adobe Photoshop as $\overline{\mathcal{S}}$.

To utilize non-differentiable image editing software $\mathcal{S}$, we train the generator using RL. In RL, an agent decides which action to execute according to the current state. We define an original image $\mathbf{x}$ as the state and the parameter vector $\mathbf{a}$ as the action. In the existing RL methods for image processing (Cao et al. 2017; Li et al. 2018; Yu et al. 2018; Furuta, Inoue, and Yamasaki 2019; Ganin et al. 2018), the agent receives operated images and decides actions sequentially, whereas our agent receives an image and selects an action only once. This is because $\mathcal{S}$ is not a linear function, for sequential actions $\mathbf{a}_1$ and $\mathbf{a}_2$,

$$\mathcal{S}(\mathbf{x}, \mathbf{a}_1 + \mathbf{a}_2) \neq \mathcal{S}(\mathcal{S}(\mathbf{x}, \mathbf{a}_1), \mathbf{a}_2). \quad (4)$$

Because it is hard for users to interpret sequential actions, we use only single-step actions.

We define the reward so that $\mathbf{y}' = \mathcal{S}(\mathbf{x}, \mathbf{a})$ cannot be distinguished from images of the target domain $Y$. The simplest reward is $D(\mathbf{y}')$, but maximizing only $D(\mathbf{y}')$ can lead to lack of consistency between $\mathbf{x}$ and $\mathbf{y}'$. To deceive the discriminator with as small a change as possible, we define the reward $R$ as follows:

$$R = D(\mathbf{y}') - \alpha \mathrm{MSE}(\mathbf{x}, \mathbf{y}'), \quad (5)$$

where the second term calculates the mean squared error between two images.

We select advantage actor-critic (A2C) (Mnih et al. 2016) as a method of RL following a training strategy by Ganin et al. (2018). A2C consists of value network $V$ and policy network $\pi$. Value network $V(\mathbf{x})$ is a module that estimates the value of the current state $\mathbf{x}$. The loss to optimize $V$ is defined as follows:

$$\mathcal{L}_V = \left(V(\mathbf{x}) - R\right)^2/2. \tag{6}$$

Policy network $\pi(a_k|\mathbf{x})$ is a module that outputs the probability of each action $a_k$ in the current state $\mathbf{x}$ and is trained to maximize the expected reward,

$$\mathcal{L}_\pi = \sum_k \left(-\log\pi\left(a_k|\mathbf{x}\right)\left(R - V\left(\mathbf{x}\right)\right) - \beta H\left(\pi\left(a_k|\mathbf{x}\right)\right)\right). \tag{7}$$

Intuitively, if the reward obtained by the operation $\mathbf{a} = [a_1, a_2, ..., a_K]$ is greater than the reward predicted by the value network, the probability of $\mathbf{a}$ increases. The second term is a function that calculates entropy, which encourages the agent to explore and prevents convergence to local optima.

## Network Architecture

In this paper, we use the discriminator and the generator whose architecture is shown in Figure 2. The discriminator has general CNN architecture similar to the one used in WGAN-GP (Gulrajani et al. 2017). The generator consists of the policy network and the value network, which share the two-dimensional (2D) convolutional layers.

$\mathcal{S}$ can take continuous parameters, but an agent which selects continuous actions is hard to train. Therefore, we design our agent to take discrete actions and the policy network to output probabilities for each discrete action. We name the output of the policy network as $\mathbf{q}$, which is a matrix of $\mathbb{R}^{L\times K}$, and $L$ is the number of the discrete steps of the parameters. $\mathcal{S}$ has a maximum value $a_k^{max}$ and a minimum value $a_k^{min}$ for each $a_k$. We divide the range between maximum and minimum values into $L$ steps, and the policy network outputs probabilities for each discrete action as follows,

$$\pi\left(a_k^{min} + \left(a_k^{max} - a_k^{min}\right) \times \frac{l-1}{L-1}\,\middle|\,\mathbf{x}\right) = q_{lk}, \tag{8}$$

where $l \in \{1, 2, ..., L\}$. To represent relationship between adjacent discrete steps (*e.g.*, $q_{lk}$ and $q_{(l+1)k}$), we use one-dimensional (1D) convolutional layers to make the probabilities $\mathbf{q}$ from the CNN feature. We do not use padding for the 1D convolutional layers, because the padding can generate strange probability values at both ends of the steps and can destabilize the training.

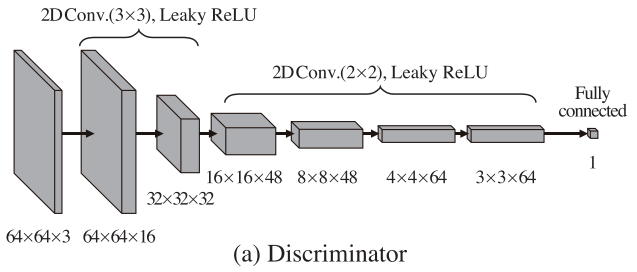
## Train and Test

While training, we resize all images to $64 \times 64$px, and select the action $\mathbf{a}$ probabilistically according to $\pi$, that is,

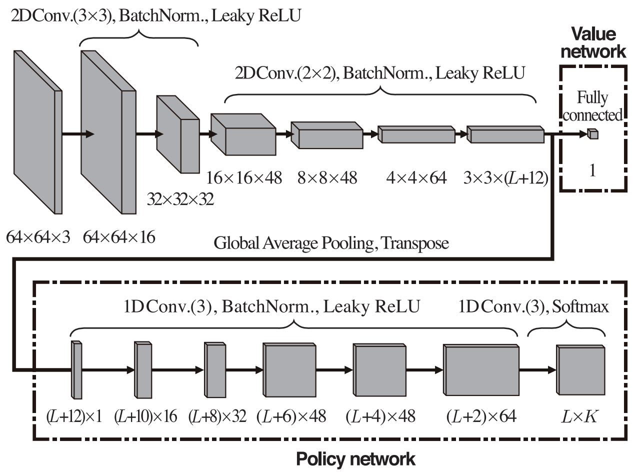
$$a_k \sim \pi(a_k|\mathbf{x}). \tag{9}$$

The resized image is edited according to $\mathbf{a}$. While testing, the agent takes an image resized to $64 \times 64$px as a state and selects action $\mathbf{a}$ deterministically,

$$l_k = \arg\max_l q_{lk}, \tag{10}$$



(a) Discriminator



(b) Generator

Figure 2: Network architecture of the discriminator and the generator.

$$a_k = a_k^{min} + (a_k^{max} - a_k^{min}) \times \frac{l_k - 1}{L-1}. \tag{11}$$

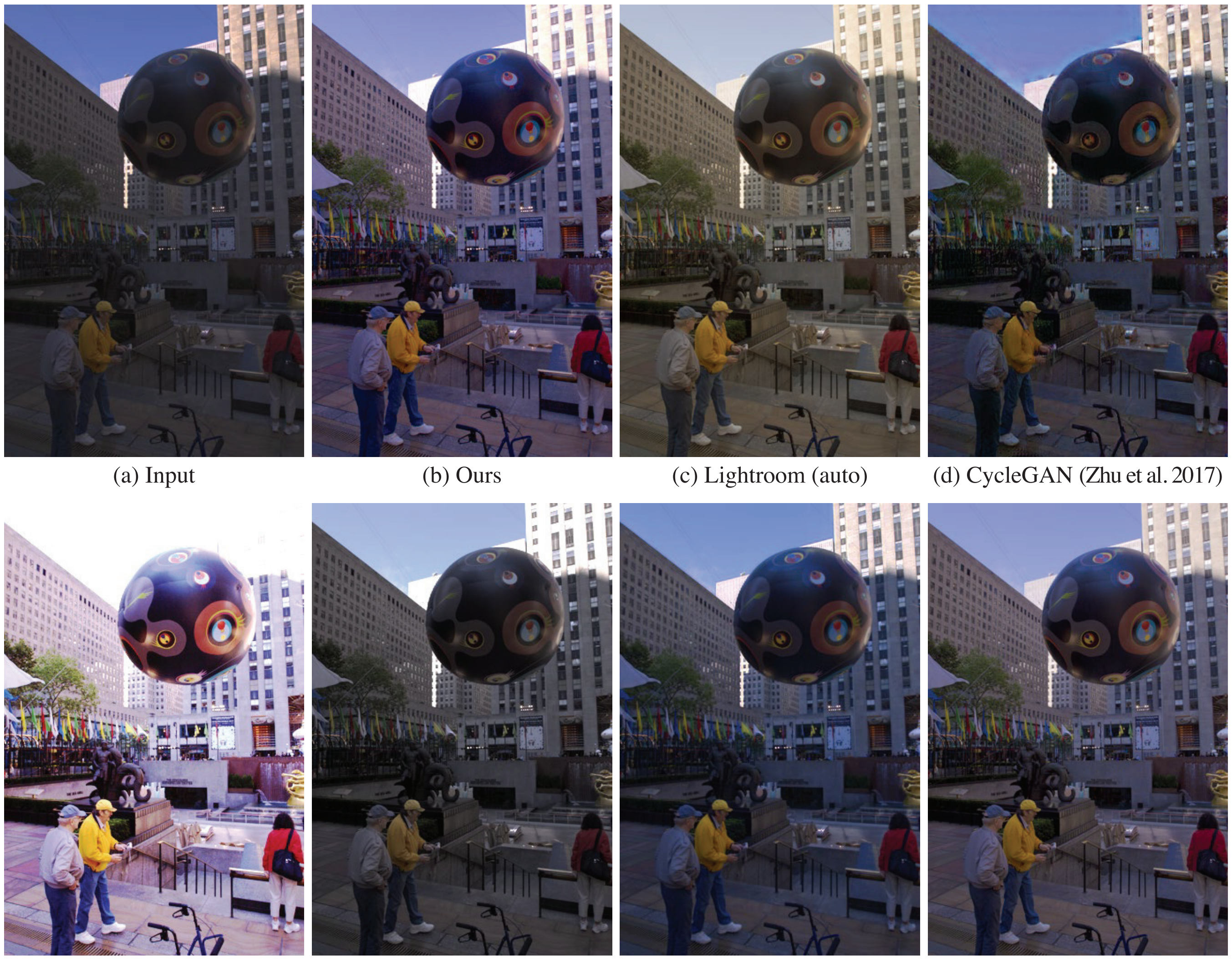
Then, the selected action $\mathbf{a}$ is applied to the original image because the operation of image editing software $\mathcal{S}$ is scale-invariant.

We train the discriminator and the generator alternately. According to the paper of WGAN-GP (Gulrajani et al. 2017), the discriminator should be updated more frequently than the generator. Following Ganin et al.'s (2018) training strategy, we create a replay buffer which keeps images generated through the training process. For every update of the generator, the discriminator is updated $U$ times using images from the replay buffer.

## Experiments

### Photo Enhancement

**Dataset** We apply the proposed method to photo enhancement, a task of converting an original photo into an expert-retouched photo. We use the MIT-Adobe 5K dataset (Bychkovsky et al. 2011) for training and testing. The dataset consists of 5,000 photos, and each image is retouched by five experts. Following Chen et al. (2018b), we use the images retouched by Expert C as the target domain images. To create unpaired image sets, we use 2,250 original images and non-overlapping 2,250 retouched images as training data, and the other 500 pairs are used as test data.


(a) Input (b) Ours (c) Lightroom (auto) (d) CycleGAN (Zhu et al. 2017)

(e) Exposure (Hu et al. 2018) (f) D&R (Park et al. 2018) (g) DPE (Chen et al. 2018b) (h) Expert-retouched

Figure 3: Qualitative comparison with other methods on a test image from the MIT-Adobe 5K dataset (Bychkovsky et al. 2011).

**Implementation** We choose Adobe Lightroom® as the image editing software $\mathcal{S}$. This tool can adjust the color, brightness, or contrast of an image by manipulating various filter parameters. From the available filters, we choose the following: Dehaze, Clarity, Contrast, Exposure, Temp, Tint, Whites, Blacks, Highlights, Shadows, Vibrance, and Saturation. Because it is difficult to use Lightroom directly, we reproduce the filters on Python. We optimize the discriminator and the generator using Adam (Kingma and Ba 2014) with a learning rate of $10^{-4}$. Other parameters $\lambda$, $\alpha$, $\beta$, $L$, and $U$ are 10, 100, 0.001, 33, and 5, respectively.

**Quantitative Evaluation** We conduct a quantitative comparison with the existing methods. We measure the difference between our result images and expert-retouched images using two common metrics, *i.e.*, PSNR and SSIM. In general, higher PSNR and SSIM mean better results. To confirm that the proposed method is scale-invariant, we conduct evaluations with small and large images whose longer sides are 512px and 2048px, respectively. We compare our method with CycleGAN (Zhu et al. 2017) and some unpaired photo enhancement methods: Exposure (Hu et al. 2018), Distort-and-Recover (D&R) (Park et al. 2018), and Deep Photo Enhancer (DPE) (Chen et al. 2018b). CycleGAN and DPE, which use CNN-based decoders, are trained using small images. When testing with large sized images, small size results are resized to large size using bicubic interpolation. D&R and Exposure, which are filter-based methods, can apply the same enhancement to small and large images.

The result of the comparison is shown in Table 1. This result shows that our method achieves the best performance for both sizes. DPE and the proposed method have almost the same values for SSIM with small size, with which the model is trained, but DPE seriously drops SSIM on large sized images because the method is not scale-invariant. D&R and Exposure are filter-based methods and perform well for large

Table 1: The result of the quantitative comparison on the MIT-Adobe 5K dataset (Bychkovsky et al. 2011).

| | Small (512px) | | Large (2048px) | |
|---|---|---|---|---|
| Method | PSNR | SSIM | PSNR | SSIM |
| Input | 19.08 | 0.823 | 18.97 | 0.814 |
| CycleGAN (Zhu et al. 2017) | 21.89 | 0.800 | 21.11 | 0.664 |
| Exposure (Hu et al. 2018) | 16.53 | 0.794 | 16.42 | 0.785 |
| D&R (Park et al. 2018) | 21.60 | 0.875 | 21.53 | 0.862 |
| DPE (Chen et al. 2018b) | 21.86 | 0.880 | 20.93 | 0.718 |
| Ours w/ Differentiable Filters | 21.06 | 0.852 | 20.94 | 0.833 |
| Ours w/o MSE | 20.50 | 0.838 | 20.59 | 0.826 |
| Ours w/o 1D Conv. | 19.31 | 0.808 | 19.34 | 0.792 |
| Ours | **22.27** | **0.881** | **22.21** | **0.868** |

Table 2: The result of the user study on the MIT-Adobe 5K dataset (Bychkovsky et al. 2011).

| Method | Average |
|---|---|
| Lightroom (auto) | 3.18 |
| CycleGAN (Zhu et al. 2017) | 3.14 |
| Exposure (Hu et al. 2018) | 3.13 |
| D&R (Park et al. 2018) | 3.22 |
| DPE (Chen et al. 2018b) | 3.28 |
| Ours | **3.42** |

images, but the filters used in these methods are simple ones, resulting in scores lower than ours. Compared to these filter-based methods, our method can use high-quality non-differentiable filters and achieve image enhancement with high performance.

To analyze our method, we conduct ablation experiments. First, we focus on the differentiability of the filters. Our proposed framework is trained with RL, which enables us to use non-differentiable filters in Lightroom. To verify that the non-differentiable filters contribute to the high performance, we replace them with differentiable filters used by Hu et al. (2018) (Ours w/ Differentiable Filters). As shown in the result, we obtain higher performance by using filters in Lightroom, and the availability of non-differentiable filters is important to the high performance.

We also conduct experiments where we remove the mean squared error from the reward (Ours w/o MSE) and replace the 1D convolutional layers with a fully connected layer (Ours w/o 1D Conv.). The results show that the mean squared error and the 1D convolutional layers are necessary factors for the high performance.

**Qualitative Evaluation** We show a qualitative comparison with the other methods for a small sample in Figure 3. In addition to the methods compared in the quantitative evaluation, we use "auto white-balance" and "auto-tone adjustment" functions available in Adobe Lightroom, which we

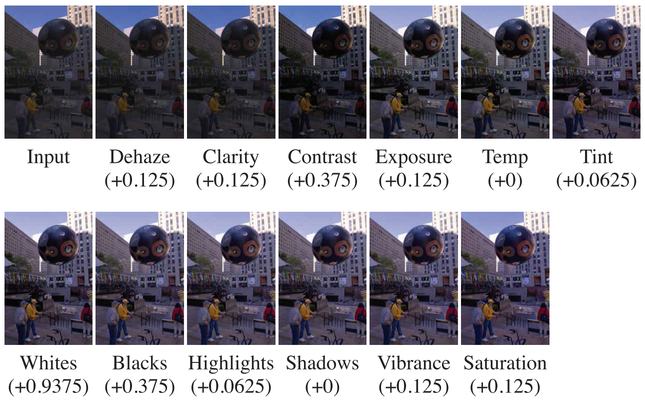


Figure 4: Application process of the filters. Values in parentheses are filter parameters, which are normalized to [-1, 1].

name *Lightroom (auto)*. As shown in this result, the Lightroom (auto) makes the color dull, CycleGAN generates artifacts at the boundary between the sky and the building, Exposure overexposes the image, and D&R outputs a slightly darker image than the target image. Compared to these methods, our method can enhance the image without any artifacts and properly reproduces the retouch by the expert. DPE can achieve almost the same quality as ours but is scale-sensitive as shown in the quantitative evaluation.

We show the sequential application process of the filters in Figure 4. Our proposed framework uses image editing software, which enables users to interpret the enhancement and manually adjust it. Note that although the filters are sequentially applied, the agent selects all filter parameters at once.

**User Study** We evaluate the proposed method through a user study. We randomly select 100 original images from 500 test pairs and perform enhancement using each existing method and the proposed method. 20 crowdworkers are hired via Amazon Mechanical Turk and presented with 100 groups of results from existing and proposed methods, which are arranged randomly to avoid bias. Then, we ask the crowdworkers to give a five-grade rating from 1 (Bad) to 5 (Excellent). Table 2 shows the average of all evaluations. Our proposed method obtains higher evaluation than all the existing methods, which shows that it is capable of a high-quality enhancement.

## Face Beautification

**Dataset** We apply the proposed method to face beautification, a task of converting a less attractive face into an attractive face. For training and testing, we use the SCUT-FBP5500 dataset (Liang et al. 2018), which has a total of 5,500 facial images and attractiveness scores within [1, 5]. We consider images with top 1,500 attractiveness scores as attractive images and the others as less attractive images. Less attractive images with the lowest 1,500 attractiveness scores and all attractive images are used for the training, and remaining less attractive images are used for the test. We extract key points using the method of Kazemi et al. (2014) to align face positions and resize images to 224×224px. The

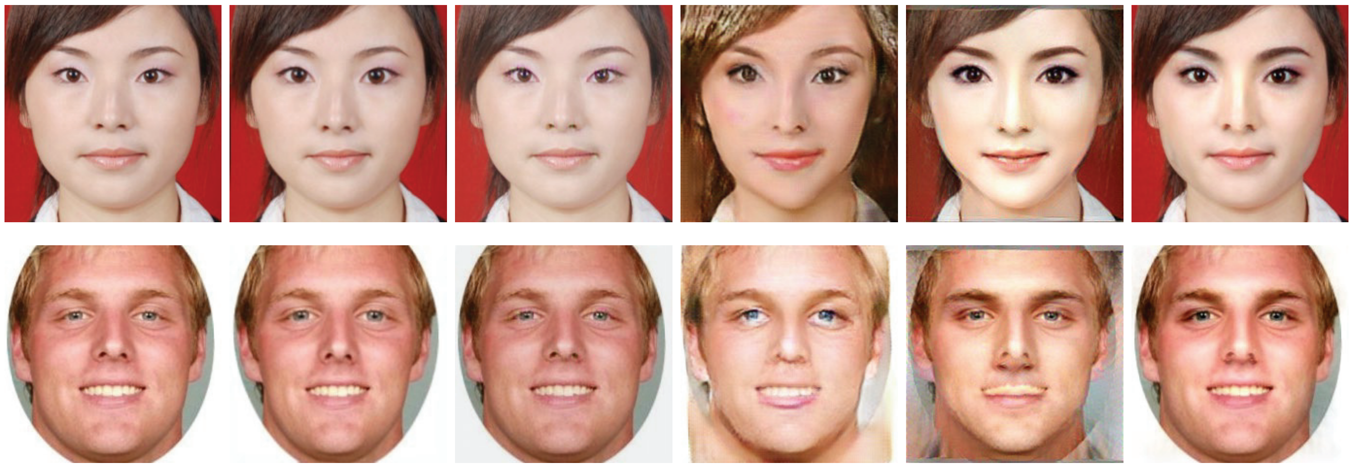

(a) Input (b) Ours (c) ResGAN (Shen and Liu 2017) (d) CycleGAN (Zhu et al. 2017) (e) DFI (Upchurch et al. 2017) (f) Facelet (Chen et al. 2018a)

Figure 5: Qualitative comparisons with other methods on test images from the SCUT-FBP5500 dataset (Liang et al. 2018).

Table 3: The result of the user study on the SCUT-FBP5500 dataset (Liang et al. 2018).

| Method | Average |
|---|---|
| ResGAN (Shen and Liu 2017) | 3.00 |
| CycleGAN (Zhu et al. 2017) | 2.74 |
| DFI (Upchurch et al. 2017) | 2.76 |
| Facelet (Chen et al. 2018a) | 2.97 |
| Ours | **3.50** |

area outside of the face is masked-out with zero value while training to remove background information.

**Implementation** For image editing software $\mathcal{S}$, we choose the Face-Aware Liquify function in Adobe Photoshop, which provides filters to morph facial images by changing geometric structure such as eye size or face contour. From the available filters, we choose the following: Eye Size, Nose Height, Nose Width, Upper Lip, Lower Lip, Mouse Width, Mouse Height, Forehead, Chin Height, and Chin Contour. Because it is difficult to use Adobe Photoshop directly, we reproduce the filters on Python. The hyperparameters are the same as those used for photo enhancement, except that $\alpha$ and $L$ are 300 and 17, respectively.

**Qualitative Evaluation** In Figure 5, we show qualitative comparisons with CycleGAN (Zhu et al. 2017) and some face attribute manipulation methods: ResGAN (Shen and Liu 2017), DFI (Upchurch et al. 2017), and Facelet (Chen et al. 2018a). All of these methods use CNN to manipulate portraits. As shown in the results, ResGAN only generates artifacts around the eyes. Although CycleGAN, DFI, and Facelet try to make the images look attractive, the edited images have artifacts derived from the structure of CNNs, which can prove fatal for the task of face beautification. Compared to these methods, our method can naturally beautify the faces by manipulating geometric structure such as enlarging the eyes or thinning the contours.

**User Study** We evaluate the proposed method by a user study. 100 images are randomly selected from less attractive images excluding those used for the training, and we perform beautification using each existing method and the proposed method. We ask crowdworkers to evaluate the images according to naturality and preference in the same way as is done for photo enhancement. Table 3 shows the average of all evaluations. The proposed method obtains higher evaluation than all existing methods, which shows that our proposed method is capable of high-quality beautification.

## Conclusions

In this study, we address unpaired image enhancement, a task of learning a mapping function which transforms input images into enhanced images in the absence of input-output image pairs. Existing CNN-based methods have the following problems: generated images have artifacts due to neural network architecture, only images with limited resolution can be generated, and the enhancement cannot be interpreted. To solve these problems, we use image editing software such as Adobe Photoshop, which can perform high-quality enhancement and avoids the problems. To use image editing software in a GAN, we propose an RL framework where the generator works as an agent controlling the software and the output of the discriminator is used as the reward. The framework can use carefully designed non-differentiable filters, which enable high-quality enhancement. We apply the proposed method to photo enhancement and face beautification. The experimental results show that our method performs better than existing methods.

## Acknowledgments

A part of this research was supported by JSPS KAKENHI Grant Number 19K22863.

## References

Bychkovsky, V.; Paris, S.; Chan, E.; and Durand, F. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR*, 97–104.

Cao, Q.; Lin, L.; Shi, Y.; Liang, X.; and Li, G. 2017. Attention-aware face hallucination via deep reinforcement learning. In *CVPR*, 690–698.

Chen, Y.-C.; Lin, H.; Shu, M.; Li, R.; Tao, X.; Shen, X.; Ye, Y.; and Jia, J. 2018a. Facelet-bank for fast portrait manipulation. In *CVPR*, 3541–3549.

Chen, Y.-S.; Wang, Y.-C.; Kao, M.-H.; and Chuang, Y.-Y. 2018b. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In *CVPR*, 6306–6314.

Chen, Y.-C.; Shen, X.; Lin, Z.; Lu, X.; Pao, I.; Jia, J.; et al. 2019. Semantic component decomposition for face attribute manipulation. In *CVPR*, 9859–9867.

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 8789–8797.

Furuta, R.; Inoue, N.; and Yamasaki, T. 2019. Fully convolutional network with multi-step reinforcement learning for image processing. In *AAAI*, 3598–3605.

Ganin, Y.; Kulkarni, T.; Babuschkin, I.; Eslami, S. A.; and Vinyals, O. 2018. Synthesizing programs for images using reinforced adversarial learning. In *ICML*, 1652–1661.

Gharbi, M.; Chen, J.; Barron, J. T.; Hasinoff, S. W.; and Durand, F. 2017. Deep bilateral learning for real-time image enhancement. In *ACM TOG*, volume 36, 118.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *NIPS*, 5767–5777.

Hu, Y.; He, H.; Xu, C.; Wang, B.; and Lin, S. 2018. Exposure: A white-box photo post-processing framework. In *ACM TOG*, volume 37, 26.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *CVPR*, 1125–1134.

Kazemi, V., and Sullivan, J. 2014. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 1867–1874.

Kim, T.; Cha, M.; Kim, H.; Lee, J. K.; and Kim, J. 2017. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 1857–1865.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, D.; Wu, H.; Zhang, J.; and Huang, K. 2018. A2-rl: Aesthetics aware reinforcement learning for image cropping. In *CVPR*, 8193–8201.

Liang, L.; Lin, L.; Jin, L.; Xie, D.; and Li, M. 2018. Scut-fbp5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction. In *ICPR*, 1598–1603.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*, 1928–1937.

Park, J.; Lee, J.-Y.; Yoo, D.; and So Kweon, I. 2018. Distort-and-recover: Color enhancement using deep reinforcement learning. In *CVPR*, 5928–5936.

Shen, W., and Liu, R. 2017. Learning residual images for face attribute manipulation. In *CVPR*, 4030–4038.

Upchurch, P.; Gardner, J.; Pleiss, G.; Pless, R.; Snavely, N.; Bala, K.; and Weinberger, K. 2017. Deep feature interpolation for image content changes. In *CVPR*, 7064–7073.

Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Tao, A.; Kautz, J.; and Catanzaro, B. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 8798–8807.

Wang, R.; Zhang, Q.; Fu, C.-W.; Shen, X.; Zheng, W.-S.; and Jia, J. 2019. Underexposed photo enhancement using deep illumination estimation. In *CVPR*, 6849–6857.

Yan, Z.; Zhang, H.; Wang, B.; Paris, S.; and Yu, Y. 2016. Automatic photo adjustment using deep neural networks. In *ACM TOG*, volume 35, 11.

Yi, Z.; Zhang, H.; Tan, P.; and Gong, M. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2849–2857.

Yu, K.; Dong, C.; Lin, L.; and Change Loy, C. 2018. Crafting a toolchain for image restoration by deep reinforcement learning. In *CVPR*, 2443–2452.

Zhang, G.; Kan, M.; Shan, S.; and Chen, X. 2018. Generative adversarial network with spatial attention for face attribute editing. In *ECCV*, 417–432.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2223–2232.