Distort-and-Recover: Color Enhancement using Deep Reinforcement Learning

Jongchan Park*[1], Joon-Young Lee[2], Donggeun Yoo[1,3], and In So Kweon[3]

[1]Lunit Inc. [2]Adobe Research [3]Korea Advanced Institute of Science and Technology (KAIST)

Abstract

Learning-based color enhancement approaches typically learn to map from input images to retouched images. Most of existing methods require expensive pairs of input-retouched images or produce results in a non-interpretable way. In this paper, we present a deep reinforcement learning (DRL) based method for color enhancement to explicitly model the step-wise nature of human retouching process. We cast a color enhancement process as a Markov Decision Process where actions are defined as global color adjustment operations. Then we train our agent to learn the optimal global enhancement sequence of the actions. In addition, we present a 'distort-and-recover' training scheme which only requires high-quality reference images for training instead of input and retouched image pairs. Given high-quality reference images, we distort the images' color distribution and form distorted-reference image pairs for training. Through extensive experiments, we show that our method produces decent enhancement results and our DRL approach is more suitable for the 'distort-and-recover' training scheme than previous supervised approaches. Supplementary material and code are available at https://sites.google.com/view/distort-and-recover/

1. Introduction

With the widespread use of digital imaging devices, photo retouching is becoming more popular. Professional photo retouching software such as Adobe Photoshop or Lightroom provides various retouching operations, but it requires expertise in photo editing and also lots of effort for achieving satisfying results. Casual software, such as Instagram on mobile platforms, provides several predefined stylization filters which make photo retouching easier for users, but the filters only perform predefined operations regardless of the context or mood of a photo and result in poor results in many cases.

Automatic color enhancement is a non-trivial task because of the highly non-linear and subjective nature of photo retouching. Consider a human professional, the retouching procedure is a sequence of iterative decision making given the image's context and color distribution. A human professional iteratively applies retouching operations until the color distribution fits the individual's taste. Therefore, the translation of pixel values to the optimal states is a complex combination of the pixel's value and the global/local color/contextual information. Also, it is a perceptual process such that there can be varying optimal states for different individuals [1, 7]. As a result, the translation from an input color distribution to the optimal becomes highly non-linear and multi-modal.

There are several lines of research for automatic color enhancement. Exemplar-based methods [4, 8, 7] tackle the non-linearity and/or multi-modality by transferring exemplar images' color distribution to the target image. During the process, exemplar images are given or retrieved from a database and the target image's color distribution is translated into the exemplar images' color distribution. In the learning-based approaches [1, 19, 20], a mapping function from the source color distribution to the target color distribution is learned from training data. Recent work [20] handles the high non-linearity of this problem with a deep neural network. The multi-modality problem is not yet explicitly solved by any learning-based method. A general problem of learning-based methods is expensive paired datasets, such as MIT-Adobe FiveK dataset [1] of input-retouched image pairs or the dataset [19] of input-action pairs.

In this paper, we propose a novel approach for automatic color enhancement with two distinctive aspects. First, we present a deep reinforcement learning (DRL) approach for color enhancement (see Figure 1). We cast the problem into a Markov Decision Process (MDP) where each step action is defined as a global color adjustment operation, such as brightness, contrast, or white-balance changes. Therefore it explicitly models iterative, step-by-step human retouching process. We solve such MDP problem with a well-studied deep reinforcement learning framework, Deep Q-Network

*This work was done while the author was at KAIST.

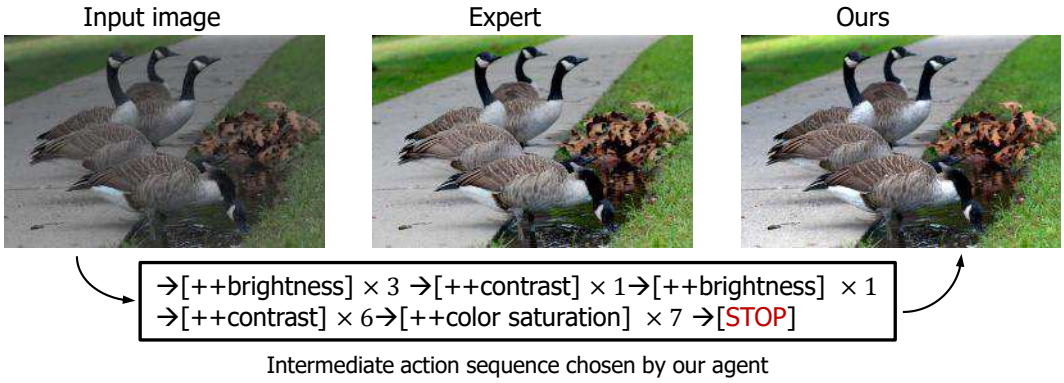

Figure 1. One example of our method. Using deep reinforcement learning, we train our agent with human expert's images in MIT-Adobe FiveK dataset [1]. The agent iteratively selects an editing operation to apply and automatically produces a retouched image with an interpretable action sequence.

(DQN) [11].

Second, we propose an economic *'distort-and-recover'* training scheme to avoid using expensive paired datasets. It is a simple and straightforward approach that only requires a set of high-quality reference images to learn color enhancement. We randomly distort the collected reference images and form distorted-reference image pairs for training. Such distorted-reference dataset is more economic than an input-retouched image set because high-quality reference images can easily be collected in personal photo albums or stock image websites. In the experiment section, we show that the *distort-and-recover* scheme is suitable for the DRL approach due to the state-exploring nature.

There are several potential applications with our method. Thanks to the economic training data collection, it is possible to train a personalized retouching agent using images retouched or selected by a user. Our experiment shows that the agents learn different styles with training datasets with different styles. In addition, if the retouching operations in our agent are defined as the operations in retouching software, then our agent can be seamlessly integrated into the retouching software.

Contributions. Our main contribution is three-fold.

1. We present a color enhancement agent that learns a step-by-step retouching process without any explicit supervision for intermediate steps.
2. We propose a *distort-and-recover* training scheme that enables us to train our agent without expensive input-retouched image pairs.
3. We show that our agent, trained with the distort-and-recover scheme, can enhance images from unknown color distributions. Through a user study, we show our agent outperforms competitive baseline methods including a recent supervised learning algorithm and an auto-retouching algorithm in a commercial software.

2. Related Work

One traditional approach for color enhancement is transferring the color of an example image to a given input image. It is originated from [13] in which the global color distribution of an input image is warped to mimic an example style. There are many subsequent works to improve this technique [3]. While this approach can provide expressive enhancement and diverse stylizations, the results highly depend on example images while providing proper exemplars is challenging. Recent works [8, 7, 4] (semi-)automate exemplar selection by image retrieval methods. Liu *et al*. [8] used a keyword-based image search to choose example images. Lee *et al*. [7] learn a content-specific style ranking using a large photo collection and select the best exemplar images for color enhancement. For pixel-wise local enhancement, Hwang *et al*. [4] find candidate images from a database then search local color enhancement operators.

Learning-based color enhancement is another dominant stream [1, 19, 20]. Bychkovsky *et al*. [1] present a number of input and retouched image pairs called MIT-Adobe FiveK, which is created by professional experts. They used this data to train a model for color and tone adjustment. Yan *et al*. [20] propose a deep learning method to learn specific enhancement styles. Given the features of color and semantic context, a deep neural network as a non-linear mapping function is trained to produce the pixel color of specific styles.

In terms of step-by-step retouching process modeling, the work in [19] is the closest to our work. In [19], Yan *et al*. used a learning-to-rank approach for color enhancement. They collected the intermediate editing actions taken in the retouching process and used this data to learn a ranking model that evaluates various color enhancements of an image. In the inference stage, this method takes various actions and evaluates the results with the ranking model to select the best action. This process is repeated to obtain

a final result. There are many differences between Yan *et al*. [19] and our method. Our method is much efficient as our agent directly selects an optimal action for each step instead of evaluating various action trials. More importantly, our method does not require step-wise action annotations, which are prohibitively expensive. Also, the action set in [19] is limited by annotated datasets while our action set does not depend on the dataset. In practice, most existing learning-based methods require at least input-retouched pairs of images for training and learn to map from known source distributions to known target distributions. The difficulty in paired data collection makes personalized enhancement difficult. Also existing datasets have strong bias to specific input distributions (*e.g*. input images in MIT-Adobe FiveK), leading to poor generalization performance. On the other hand, we show that our method can be trained solely with reference images or retouched images and learn to map to target distributions from unknown input distributions.

Recently, generative adversarial networks show impressive performance on many challenging tasks. Using such network, Pix2Pix [5] translates an image to other image and has shown strong potential to the color enhancement application. It turns day images to night images or gray images to color images, and we believe this network can be applied to the color enhancement task where input images are transformed to retouched images. In the experiment, we show that Pix2Pix achieves color enhancement performance close to the state-of-the-art [20]. Setting Pix2Pix as a baseline, our method outperforms Pix2Pix under the '*distort-and-recover*' training scheme according to our user study results in Section 6.4.

3. Problem Formulation

A human expert enhances images by applying a set of color adjustment operations. To imitate the process, we formulate color enhancement as a problem for finding an optimal sequence of adjustment actions as follows.

We enhance an input image $\mathcal{I}$ by iteratively applying an adjustment action $\mathcal{A}$. We represent an image $\mathcal{I}(t)$ at a step t using a contextual feature $\mathcal{F}_{\text{context}}(I(t))$ and a global color feature $\mathcal{F}_{\text{color}}(I(t))$. Our agent determines a color adjustment action $\mathcal{A}(t)$ at each step t under the policy Ω_Θ. Therefore, our goal is to find an optimal sequence of color adjustment actions $\mathcal{T}\{\mathcal{A}_{\text{optimal}}(t) \subset \boldsymbol{\mathcal{A}}\}$ that gives the best improvement of image color.

To solve this problem, we need a metric to measure the color aesthetic of an image $\mathcal{I}(t)$. The color aesthetic depends on not only the semantic context of the image but also individual preference, therefore it is difficult to define an absolute aesthetic score function. There has been a long line of works to assess the aesthetic scores of images as summarized in [2], but there is still no winning method that gives satisfying results. Recently, deep learning based works [9, 10, 6] lead large improvements on the problem, so we initially considered representative networks as our color aesthetic metric but the results were not stable enough for our test images. Instead, to measure the color aesthetic of an image, we simply consider human retouched images $\mathcal{I}_{\text{target}}$ as ground truth and define our aesthetic metric as a negative ℓ_2 distance between an image $\mathcal{I}(t)$ and corresponding retouched image $\mathcal{I}_{\text{target}}$, which is also used in [20, 4]. Now, our goal is to find an optimal sequence of color adjustment actions $\mathcal{T}\{\mathcal{A}_{\text{optimal}}(t) \subset \boldsymbol{\mathcal{A}}\}$ that minimizes $\|\mathcal{I}(t_{final}) - \mathcal{I}_{\text{target}}\|^2$.

This formulation can be regarded as a Markov Decision Process, where the state $\mathcal{S}$ is a combination of the contextual feature and the color feature $(\mathcal{F}_{\text{context}}, \mathcal{F}_{\text{color}})$, the action space is the set of color adjustment operations $\boldsymbol{\mathcal{A}}$, and the immediate reward is the change in ℓ_2 distance as:

$$\mathcal{R}(t) = \|\mathcal{I}_{\text{target}} - \mathcal{I}(t-1)\|^2 - \|\mathcal{I}_{\text{target}} - \mathcal{I}(t)\|^2 . \quad (1)$$

An agent, parameterized by Θ, determines the policy Ω_Θ. Unlike the typical Markov Decision Process setups, there is no transitional probabilities since the operation for each color adjustment action is deterministic.

To solve this decision-making problem, we train the agent to approximate the action value $\mathcal{Q}(\mathcal{S}(t), \mathcal{A})$ and choose an action $\mathcal{A}$ that maximizes the value. The action value of an action $\mathcal{A}$ at time t is an expected sum of future rewards as:

$$\mathcal{Q}(\mathcal{S}(t), \mathcal{A}) = E\left[r(t) + \gamma \cdot r(t+1) + \gamma^2 \cdot r(t+2) + \cdots\right],$$

where γ is a discount factor. The state space, composed of the context feature and the color feature, is continuous so the complexity of the approximation is high. We therefore employ a deep neural network as an agent to handle such a high complexity. This agent estimates the action value $\mathcal{Q}(\mathcal{A}, t)$ for $\mathcal{S}(t)$.

4. Automatic Color Enhancement

Figure 2 illustrates the overall pipeline of our automatic color enhancement algorithm. To enhance the color of an input image, our method proceeds as follows. An input image $\mathcal{I}$ is forwarded to the feature extractor to extract features $\mathcal{S}$=$(\mathcal{F}_{\text{context}}, \mathcal{F}_{\text{color}})$. The agent then takes the features and estimates the action value $\mathcal{Q}(\mathcal{S}, \mathcal{A})$ for each predefined adjustment action $\mathcal{A}$. The agent then chooses an action that has the highest action value and applies the action to the input image if the value is positive. The agent repeats this process and stops when all estimated action values are negative.

4.1. Actions

As commercial photo retouching software provides various built-in actions, we also define several editing actions

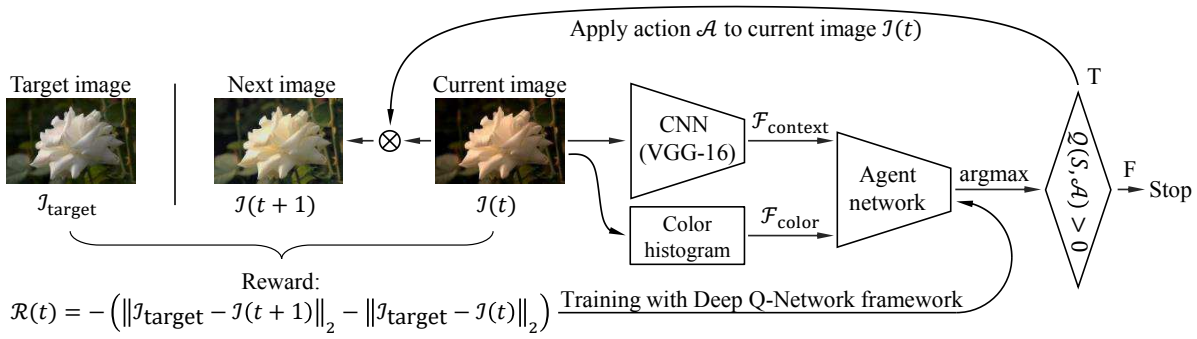

Figure 2. **Overview of our step-wise automatic color enhancement method.** Given an image $\mathcal{I}(t)$ at a sequential adjustment step t, we extract a contextual feature $\mathcal{F}_{\text{context}}$ with a pre-trained CNN and a color feature $\mathcal{F}_{\text{color}}$. These features are forwarded to the agent network, then the agent determines an optimal action $\mathcal{A}$ to make the image look better. We apply the action to the current image, and repeat this process until the agent produces a "stop" signal. Under the Deep Q-Network framework, this agent is trained to maximize the reward defined as a pixel-level distance between an input and a target retouched image.

#	Action description	#	Action description
1	↓ contrast (×0.95)	2	↑ contrast (×1.05)
3	↓ color saturation (×0.95)	4	↑ color saturation (×1.05)
5	↓ brightness (×0.95)	6	↑ brightness (×1.05)
7	↓ red and green (×0.95)	8	↑ red and green (×1.05)
9	↓ green and blue (×0.95)	10	↑ green and blue (×1.05)
11	↓ red and blue (×0.95)	12	↑ red and blue (×1.05)

Table 1. **Actions for automatic color enhancement.** We define 12 actions to adjust contrast, saturation, brightness, and whit-balance. Each action increases or decreases the value by 5%.

as shown in Table 1. We take a part of actions in [19] and add additional actions for white-balancing. In our action items, Actions of 1 and 2 are for adjusting image contrast, and Actions of 3 and 4 are for controlling color saturation. Image brightness is controlled by Actions of 5 and 6. For white-balancing, we define Actions from 7 to 12, which increase or decrease two color components. The actions are quantized since DQN [11] accepts discrete actions.

4.2. Features

The agent determines which action to take based on information embedded in features. It is important to define features with information that a human considers for photo retouching, such as color distribution and semantic context. A person's preferred color distribution of an image highly depends on the semantic context of an image, therefore contextual information is an important cue to determine the color of an image. For this reason, Hwang *et al*. [4] adopt GIST [17] as a contextual feature, Bychkovsky *et al*. [1] utilize face detection results to define a contextual feature, and Yan *et al*. [20] utilize a semantic label map estimated from scene parsing and object detection algorithms. However, GIST is not sufficient to encode high-level semantic information and relying on separate algorithms for object detection or segmentation is not efficient.

Instead, we utilize the intermediate activations of a deep convolutional neural network pre-trained on the ILSVRC classification dataset [14]. Since the network is trained to recognize 1,000 object classes with 1.3M images, intermediate activations of such network embed semantic information of an image. As studied by [15, 21], the use of the activations as a generic image feature significantly improves other visual recognition tasks. We choose the 4,096-dimensional activations from the sixth layer of VGG-16 model [16] as our contextual feature $\mathcal{F}_{\text{context}}$.

For color features $\mathcal{F}_{\text{color}}$, we adopt a CIELab color histogram. We linearly quantize each axis of the CIELab space to 20 intervals and count the number of pixels fall into each interval to obtain a 20×20×20-dimensional histogram. In addition to this feature, we have tried various types of color features, but CIELab histogram shows the best performance (see Section 6.2).

4.3. Agent

The agent takes the features extracted from the current image and estimates the action value for each of actions. The action value $\mathcal{Q}(\mathcal{S}(t), \mathcal{A})$ of an action $\mathcal{A}$ is the expected sum of future rewards with the action. Our agent consists of a 4-layer multi-layer perceptron and all layers use a rectified linear unit as activation functions.

During training, the policy Ω_Θ is determined with an ϵ-greedy algorithm. The ϵ-greedy algorithm randomly samples actions with a probability of ϵ and chooses the actions with the highest expected return in a greedy manner with a probability of $1-\epsilon$. The agent is trained with the intermediate reward $\mathcal{R}(t)$ of Equation 1 which is defined as the change of the negative ℓ_2 distance. In the inference stage, we determine the policy with a pure greedy algorithm where ϵ is zero, *i.e.* the highest expected return is always chosen. The process is repeated until all expected returns are negative.

5. 'Distort-and-Recover' Training Scheme

Despite the successful color enhancement results reported in [1, 19, 20], the learning-based methods are heavily dependent on datasets by nature. All the previous learning-based approaches require at least the input-retouched image pairs for training. Paired datasets are expensive in terms of data collection, thus it makes difficult to develop a system to account for personalized image enhancement. More importantly, such datasets often cover specific input distributions only and it leads to poor generalization due to large distribution changes of inputs (*e.g.* when an iPhone image is tested with a color retouching model trained with raw input images on MIT-Adobe FiveK [1]).

To resolve the problem, we propose a '*distort-and-recover*' training scheme, which only utilizes retouched or curated reference images. We distort high-quality reference images by randomly applying color adjustment operations and synthesize a *pseudo* input-retouched pairs of images. In order to deliver clearer supervisory signals with efficient search space, the $\mathcal{L}^2$ distance from the distorted image to the reference image is kept from 10 to 20 in the CIELab colorspace. Also, to prevent the bias of color distortion, we use different global operations from the DRL agent's action set: brightness/contrast/color saturation adjustments in highlight/shadow pixels, and C/M/Y/R/G/B adjustments in highlight C/M/Y/R/G/B pixels respectively. In selecting highlight/shadow pixels, we use a soft pixel selection method with a variant of sigmoid function which applies high weights on pixels with high/low values. Other than the highly non-linear operations, we also use basic brightness/contrast/color saturation adjustments. Distortion operations are designed with simplicity in mind. Any type of global operations can be used for distortion. Details about the random distortion operations are shown in the supplementary materials.

One may argue that there is expensive human labor included in the retouched reference images, but in the aspect of data collection, retouched reference images are already openly available in many areas. For example, high-quality reference images can be collected from stock image websites, Flickr, or Aesthetic Visual Analysis dataset [12]. Input-retouched image pairs, however, are not usually openly available thus creating a new paired dataset requires intensive labor of professionals.

We will validate the effectiveness of the '*distort-and-recover*' training scheme in Section 6.4. Surprisingly, our DRL agent works well with the synthesized pairs while a supervised learning method performs poorly.

6. Experiments

In this section, we validate our method through extensive experiments: 1) evaluation with different sets of features as input to the agent network in Section 6.2, 2) evaluation on input-retouched paired datasets in Section 6.3), 3) evaluation of the "*distort-and-recover*" training scheme with a user study in Section 6.4.

6.1. Experimental Setup

We have built our network with Tensorflow[1]. Among various follow-up studies of DQN, we adopt the Double Q-Learning method [18] for deep reinforcement learning. Our agent network consists of 4 fully-connected layers, the output size of the layers are 4096, 4096, 512, 12, and the input size may vary depending on the type of input features in use. It takes at least 12 hours to train the agent network on a single NVIDIA GTX 1080 with MIT-Adobe FiveK dataset. The duration varies depending on the size of training set. The mini-batch size is 4, the base learning rate is 10^{-5}, the minimum learning rate is 10^{-8}, and learning rate decays by a factor of 0.96 every 5,000 iterations. Among many default optimizers provided in Tensorflow, for our task, Adam optimizer shows the most stable training with the best result.

Pix2Pix Baseline. The current state-of-the-art method in automatic color enhancement is [20], but it is hard to directly compare results[2]. Instead, we use Pix2Pix [5] as a strong baseline for a pixel-level prediction method. Pix2Pix is a conditional generative adversarial network that translates images into a different domain, and we believe it is a general and advanced technique for image generation. Thus, Pix2Pix is applicable to the color enhancement problem.

For fair comparisons, we tune the hyperparameters of Pix2Pix throughout extensive trial and errors to get the best enhancement results. The input of the Pix2Pix implementation [3] is 256×256 squared images. Direct warping to squared images may distort the color distribution of the images, so we keep the aspect ratio and add zero-pixels to form squared input-output pairs and then resize them into 256×256 resolution. The input channels are changed from the RGB to CIELab colorspace in which pixels are re-scaled to [-1, +1], and the RGB $\mathcal{L}^1$ loss is changed to the CIELab $\mathcal{L}^2$ loss.

Dataset and metrics. MIT-Adobe FiveK dataset [1] is mainly used for our experiments. It consists of 5,000 raw images, each of which has paired with retouched images from five different experts – A/B/C/D/E, so there are five sets of 5,000 input-retouched paired images. If not otherwise specified, as our test set, we use raw input images

[1] https://www.tensorflow.org/

[2] The main code of [20] is available, but we fail to get reasonable results due to the external dependencies on core modules such as the semantic label map. The full set of final results is not openly available.

[3] https://github.com/affinelayer/Pix2Pix-tensorflow

Features	mean $\mathcal{L}^2$ error
Context + RGB-L histogram	12.62
Context + TinyImage	15.07
Context + Six features in [1]	12.53
Context + CIELab histogram	**10.99**
Lab histogram	12.30

Table 2. **Comparison of mean $\mathcal{L}^2$ error** with different feature combinations on RANDOM 250. We use six features as in [1].

Method	mean $\mathcal{L}^2$ error	SSIM
Input image	17.07	-
Exemplar-based (Hwang *et al*. [4])	15.01	-
DeepNet-based (Yan *et al*. [20])	**9.85**	-
Pix2Pix [5]	10.49	0.857
Ours	10.99	**0.905**

Table 3. **Comparison of mean $\mathcal{L}^2$ error** in different approaches on RANDOM 250. All approaches are trained with input-retouched image pairs by Expert C on MIT-Adobe FiveK.

on RANDOM 250 which is a subset of MIT-Adobe FiveK dataset randomly selected in [4], and the rest of 4,750 images are used for training. For quantitative comparison, we use the mean $\mathcal{L}^2$ error metric to evaluate the performance, following the previous approaches [4, 20].

6.2. Feature Selection

In this section, we compare the results from various combinations of context and color features as the input to the agent network. As described in Section 4.2, we use the convolutional feature at the sixth layer of VGG16 [16] model pretrained on ImageNet [14] as our contextual feature. Together with the contextual feature, we evaluate various color features, since deep features contain high-level semantic information while the low-level information is important for color enhancement.

We train the agent network with different sets of features. We use the input-retouched image pairs by Expert C on MIT-Adobe FiveK and measure the mean $\mathcal{L}^2$ error for quantitative comparison. All other setups are identical.

Table 2 demonstrates that the combination of the CIELab color histogram and the contextual feature yields the best performance in terms of mean $\mathcal{L}^2$ error on RANDOM 250. It is also verified that the contextual feature plays an important role for color enhancement as the error increases a lot without the contextual feature. All experiments in the following sections, we use the best feature combination, the VGG feature and the CIELab color histogram.

6.3. Input-Retouched Dataset

We compare our method with the previous methods [4, 20] by learning a specific human expert's color adjustment style. Training setup is the same as Section 6.2.

Table 3 summarizes the results of different methods and Figure 3 shows example results for qualitative comparison.

Method	mean $\mathcal{L}^2$ error	SSIM
Input image	17.07	-
Pix2Pix [5] - no augmentation	14.46	0.699
Pix2Pix [5] - augmented 10 times	13.79	0.804
Pix2Pix [5] - augmented 20 times	13.59	0.786
Pix2Pix [5] - augmented 30 times	13.60	0.772
Ours - no augmentation	**12.15**	**0.910**

Table 4. Mean $\mathcal{L}^2$ error and SSIM of different approaches trained with the *distort-and-recover* scheme. Retouched images by Expert C on MIT-Adobe FiveK are used as reference images.

In Table 3, our method performs better than [4] and is comparable to [5, 20]. Pix2Pix [5] is also competitive with [20]. Pixel-level prediction methods, such as [5, 20], performs well in terms of mean $\mathcal{L}^2$ error because local color enhancement provides much flexible mapping while our method is mainly limited by the predefined actions in Table 1.However, pixel-level prediction methods suffer from local artifacts especially in handling high-resolution images. To quantitatively compare these artifacts, we have measured SSIM scores in both Table 3 and Table 4, and the scores demonstrate that the superior image quality of our results. Also, compared to the network directly estimating the pixel values, our method is *interpretable* since our agent actively chooses and applies a sequence of human-defined editing operations. These intermediate retouching actions provide better understanding of the retouching process and can be useful in many scenarios like user-guided photo retouching and personalized tutorial generation. In addition, our DRL approach outperforms other methods in the proposed *distort-and-recover* scheme, which will be addressed in the following Section 6.4.

6.4. Distort-and-Recover Training

We perform an experiment to evaluate our *distort-and-recover* training scheme as described in Section 5. In this experiment, we assume that we only have high-quality reference images, not input-retouched pairs. Instead, we distort reference images by applying random actions and use the distorted-reference pairs for training.

Comparison with supervised learning We use the images retouched by Expert C in MIT-Adobe FiveK as reference images and distort them as inputs to train our agent. We also train Pix2Pix [5] with the same *distort-and-recover* training scheme. Table 4 shows the comparison result. At first, we train our agent with 4,750 pairs of distorted-reference images where only one randomly distorted image is generated for each reference image. Our approach shows decent performance with this setting, although the color distribution of raw test images are never shown to the agent network. Examples of distorted images are included in the supplementary material. In contrast, the performance

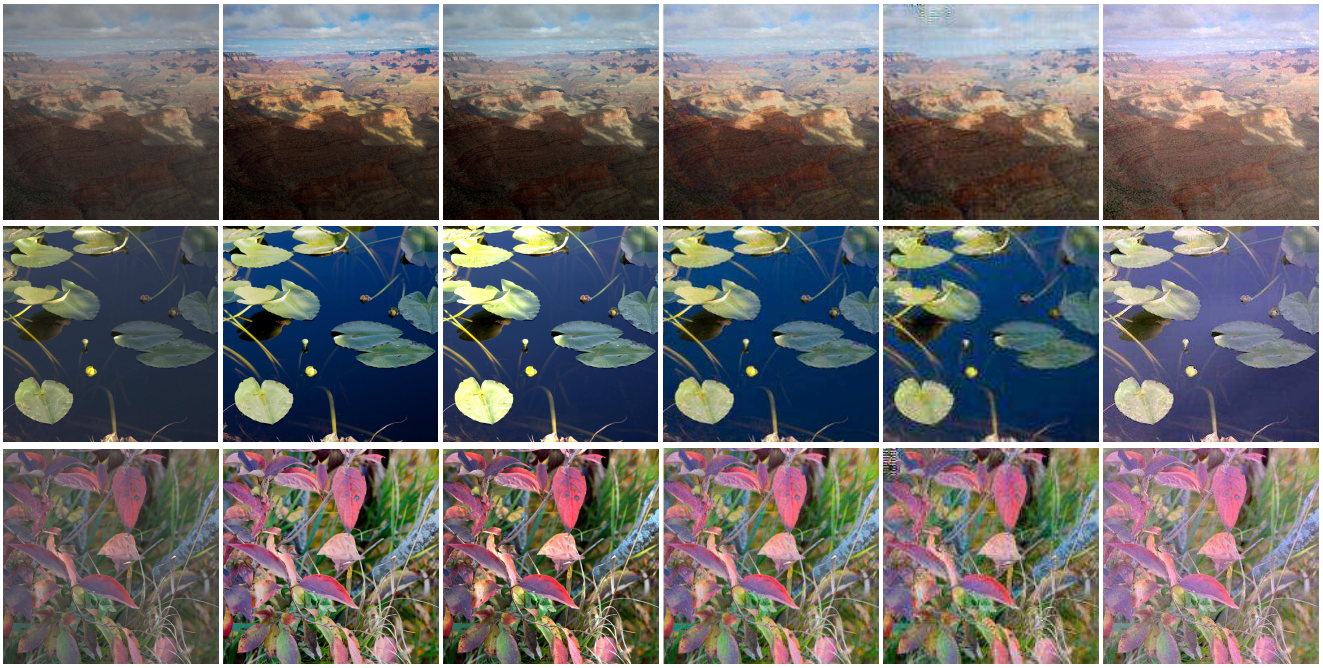

Figure 3. Qualitative comparison with Yan *et al*. [20], Pix2Pix [5], Hwang *et al*. [4] on RANDOM 250. All approaches are trained with input-retouched pairs by Expert C in MIT-Adobe FiveK. Images for Yan *et al*. [20] and Hwang *et al*. [4] are taken from the original paper and provided by the original author, respectively.

of Pix2Pix drops severely, compared to the previous result with input-retouched pairs in Table 3. When the large number of augmented pairs, in which a reference image is associated with multiple distorted images, are used for training, the performance of Pix2Pix gets better but is quickly saturated.

This experiment shows that our DRL approach works well with the *distort-and-recover* training scheme even without extra data augmentation, while supervised learning methods may suffer from unknown source color distributions. We conjecture the high performance of our approach is due to the efficient state-exploring nature of reinforcement learning: In the DQN framework, the search space is constrained with the discretized action set, and Markov property enables the agent to efficiently search the near color state. On the other hand, Pix2Pix can be seen as one-step action that translates input distribution to reference distribution with a high non-linearity in a large search space. Its performance depends highly on the training set. That is, if a training set covers limited input distributions, then the trained network suffers from poor generalization in test cases. Therefore, our DRL agent is robust to the input distribution shift while Pix2Pix is not.

Learning a style filter We perform an experiment to learn certain styles with the *distort-and-recover* scheme. We apply a predefined filter (Nashville in Instagram) to the retouched images by Expert C to create a set of reference images in a new style. We train our DRL agent with distorted and Nashville-filtered image pairs and test the agent with the raw input images on RANDOM 250. The mean $\mathcal{L}^2$ error between the inputs and corresponding ground truth images is 27.78, while the mean error between our outputs and the groundtruth images is 17.87. Qualitative results are shown in the supplementary material. These results verifies that our agent is able to learn difference retouching styles only with desired style images, demonstrating the possibility of personalized enhancement from a user-curated dataset with our *distort-and-recover* training scheme.

Distort-and-Recover with Shutterstock 150K To build an automatic color enhancement system that handles a wide range of image content, it is essential to have a large number of training data with diverse context. Our *distort-and-recover* scheme gives a practical solution to this problem. As an example case study, we collect high-quality reference images from a stock image website, Shutterstock[4]. We have crawled 150K thumbnail images from Shutterstock by searching general keywords of "food, flower, nature, portrait, mountain", sorted by popularity. Then, we train our DRL agent with the crawled reference images. The crawled images are retouched or curated by various photographers, so the images have no specific style or personal preference. From such reference images, we can expect that the agent

[4]https://www.shutterstock.com/

Input A B C D E Ours

Figure 4. Qualitative comparison between five human experts' retouching (A,B,C,D,E) and our result on MIT-Adobe FiveK. We train our agent with 150K reference images from Shutterstock using the *distort-and-recover* training scheme.

network learn a generally pleasing style according to image contents. We test our agent network for the raw images on MIT-Adobe FiveK as inputs. Figure 4 shows examples of our enhancement results compared with human experts.

We evaluate the performance of our method through a user study. For the user study, we compare the retouching results from our agent with four different sets: 1) raw input images as a baseline, 2) Pix2Pix as one of state-of-the-art supervised learning methods, 3) *Lightroom* – a commercial photo retouching software, 4) expert-retouched images as groundtruth. During training the Pix2Pix model, we observed that the model diverges and produces poor results with a large number of distorted-reference pairs, so we trained Pix2Pix with 10K distorted-reference image pairs. For Lightroom, we have retouched images by applying both 'auto white-balance' and 'auto-tone adjustment' functions in Adobe Lightroom. We prepared 'expert-retouched' images by randomly selecting one expert result out of five corresponding expert-retouched images for each input.

The user study is conducted on the Google Forms platform [5]. For each question, randomly shuffled five images are shown to the respondents. Respondents are told to give scores ranging from 0 (worst) to 5 (best) for each given image. There are 50 question sets of images, 30 respondents, and a total of 7500 scores are rated. We recruited respondents from a college campus and paid them accordingly. All the images used in the survey are randomly selected from RANDOM250 with other corresponding results. The results are listed in the supplementary materials.

Table 5 shows the result of the user study. We report 'mean scores' and 'normalized mean scores' with standard deviations. Since each respondent has different sense of score scale due to the subject manner of a problem, in 'normalized mean scores', we normalize scores per respondent so that each respondent's answers have zero mean and unit variance distribution. In the user study, our method significantly outperformed Pix2Pix and also shows superior performance than the results of Lightroom, one of the best commercial retouching software which is highly optimized for photo retouching application.

Compared with the quantitative evaluation using input-retouched pairs, Pix2Pix produced poor results and got low scores in this user study, while our method works robustly in both cases. We note that we used distorted images for training and raw images for testing in our user study, therefore we conjecture that the performance drop of Pix2Pix comes from the gap between color distributions of training and test images. More specifically, in the input-retouched pairs, the test input distributions and the train input distributions are expected to be similar, so fully supervised networks are able to learn mappings from input distributions to target distributions; in the distorted-reference pairs, however, training inputs are generated with random distortions, therefore networks should handle mappings from unknown distributions to target distributions. We empirically observe that our method performs well under such circumstances and we argue that it is due to the state-exploring nature of deep reinforcement learning.

[5] https://docs.google.com/forms/

Retouch method	Mean score	Mean score (normalized)
Input image	2.11 (±1.20)	-0.43 (±0.90)
Pix2Pix [5]	2.09 (±1.31)	-0.44 (±0.97)
Lightroom	2.74 (±1.31)	0.09 (±0.93)
Ours	**2.86** (±1.22)	**0.19** (±0.88)
Expert	3.37 (±1.28)	0.60 (±0.92)

Table 5. **User study results.** Pix2Pix [5] and ours is trained with Shutterstock 150K images using the *distort-and-recover* training scheme, and the test images are input images from MIT-Adobe FiveK dataset. The *normalized* scores are computed after normalizing scores to zero mean and unit variance for each respondent.

7. Conclusion

We have proposed a novel method for color enhancement. With the deep reinforcement learning approach, we have explicitly modeled a human retouching process and estimated interpretable retouching steps from predefined actions without extra annotations. To resolve the difficulty of collecting training data, we also have presented the *distort-and-recover* training scheme, which enables us to learn our model with large-scale data in an economic way. We have shown that our DRL approach compensates the lack of generality in distorted images effectively and is generalized well to unknown input distribution in test time. Our DRL-based color enhancement with data efficient training scheme has shown a new possibility in interesting applications including automatic tutorial generation, and personalized enhancement, and we hope to explore these applications in the future.

Acknowledgement

This work was supported by the Technology Innovation Program (No. 10048320), funded by the Ministry of Trade, Industry & Energy (MI, Korea).

References

[1] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 1, 2, 4, 5, 6

[2] Y. Deng, C. C. Loy, and X. Tang. Image aesthetic assessment: An experimental survey. *CoRR*, abs/1610.00838, 2016. 3

[3] H. S. Faridul, T. Pouli, C. Chamaret, J. Stauder, A. Trémeau, E. Reinhard, et al. A survey of color mapping and its applications. In *Eurographics (State of the Art Reports)*, pages 43–67, 2014. 2

[4] S. J. Hwang, A. Kapoor, and S. B. Kang. *Context-Based Automatic Local Image Enhancement*, pages 569–582. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 1, 2, 3, 4, 6, 7

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. 3, 5, 6, 7, 8

[6] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes. Photo aesthetics ranking network with attributes and content adaptation. In *European Conference on Computer Vision*, pages 662–679. Springer, 2016. 3

[7] J.-Y. Lee, K. Sunkavalli, Z. Lin, X. Shenand, and I. S. Kweon. Automatic content-aware color and tone stylization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[8] Y. Liu, M. Cohen, M. Uyttendaele, and S. Rusinkiewicz. Autostyle: Automatic style transfer from image collections to users' images. In *Computer Graphics Forum*, volume 33, pages 21–31. Wiley Online Library, 2014. 1, 2

[9] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang. Rapid: Rating pictorial aesthetics using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 457–466. ACM, 2014. 3

[10] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang. Deep multi-patch aggregation network for image style, aesthetics, and quality estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 990–998, 2015. 3

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015. Letter. 2, 4

[12] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2408–2415. IEEE, 2012. 5

[13] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21(5):34–41, Sept. 2001. 2

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4, 6

[15] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014. 4

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4, 6

[17] A. Torralba. Contextual priming for object detection. *International journal of computer vision*, 53(2):169–191, 2003. 4

[18] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016. 5

[19] J. Yan, S. Lin, S. B. Kang, and X. Tang. A learning-to-rank approach for image color enhancement. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2987–2994. IEEE, 2014. 1, 2, 3, 4, 5

[20] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu. Automatic photo adjustment using deep neural networks. *ACM Trans. Graph.*, 35(2):11:1–11:15, Feb. 2016. 1, 2, 3, 4, 5, 6, 7

[21] D. Yoo, S. Park, J.-Y. Lee, and I. So Kweon. Multi-scale pyramid pooling for deep convolutional representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 71–80, 2015. 4