Exposure: A White-Box Photo Post-Processing Framework

YUANMING HU, Massachusetts Institute of Technology
HAO HE, Massachusetts Institute of Technology
CHENXI XU, Peking University
BAOYUAN WANG, Microsoft Research
STEPHEN LIN, Microsoft Research

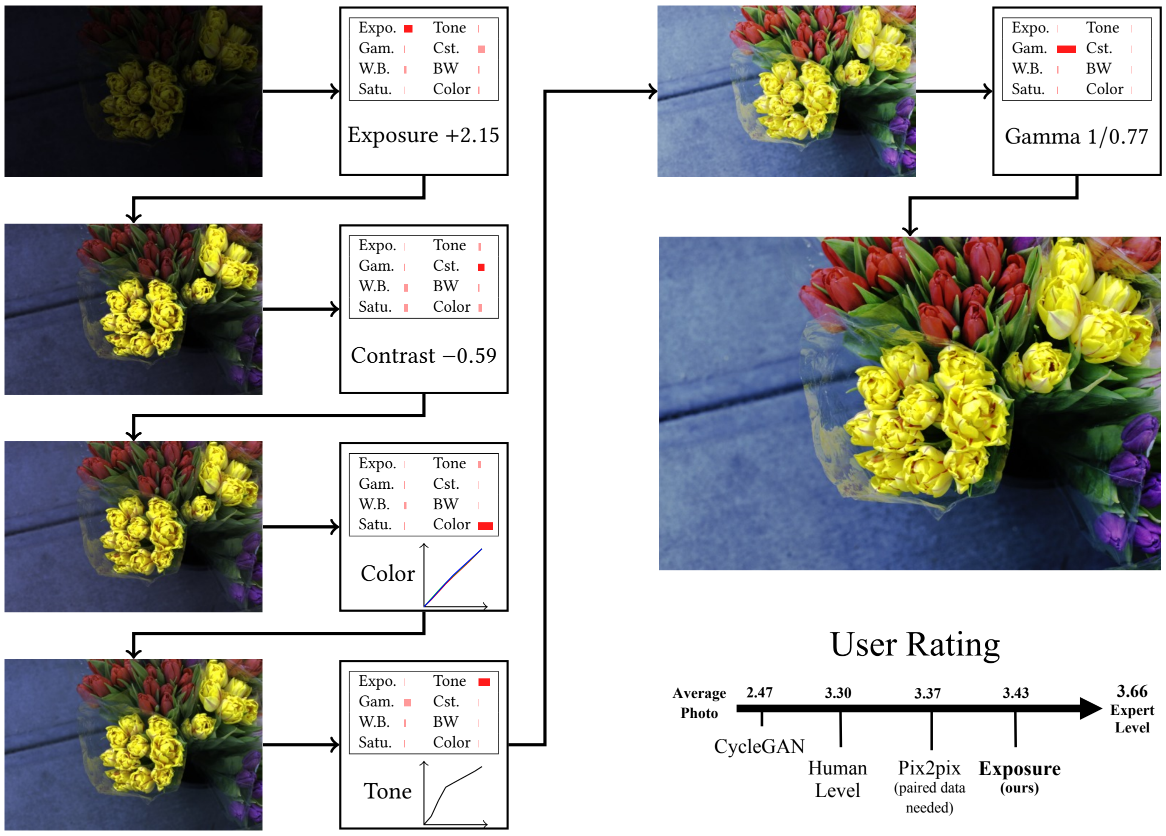


Fig. 1. Our method provides automatic and end-to-end processing of RAW photos, directly from linear RGB data captured by camera sensors to visually pleasing and display-ready images. Our system not only generates appealing results, but also outputs a meaningful filtering sequence. User studies indicate that the retouching results surpass those of strong baselines, even though our method uses only unpaired data for training.

Retouching can significantly elevate the visual appeal of photos, but many casual photographers lack the expertise to do this well. To address this problem, previous works have proposed automatic retouching systems based on supervised learning from *paired* training images acquired before and after manual editing. As it is difficult for users to acquire paired images that reflect their retouching preferences, we present in this paper a deep learning approach that is instead trained on *unpaired* data, namely a set of photographs that exhibits a retouching style the user likes, which is much easier to collect. Our system is formulated using deep convolutional neural networks that learn to apply different retouching operations on an input image. Network training with respect to various types of edits is enabled by modeling these retouching operations in a unified manner as resolution-independent differentiable filters. To apply the filters in a proper sequence and with suitable parameters, we employ a deep reinforcement learning approach that learns to make decisions on what action to take next, given the current state of the image. In contrast to many deep learning systems, ours provides users with an understandable solution in the form of conventional retouching edits, rather than just a "black box" result. Through quantitative comparisons and user studies, we show that this technique generates retouching results consistent with the provided photo set.

1 INTRODUCTION

The aesthetic quality of digital photographs can be appreciably enhanced through retouching. Experienced photographers often perform a variety of post-processing edits, such as color adjustment and image cropping, to produce a result that is expressive and more visually appealing. Such edits can be applied with the help of software such as *Photoshop* and *Lightroom*; however, photo retouching remains challenging for ordinary users who lack the skill to manipulate their images effectively. This problem underscores the need for automatic photo editing tools, which can be helpful even to professionals by providing a better starting point for manual editing.

An important consideration in photo retouching is that different people have different preferences in retouching style. While some people like photos with vibrant colors that pop out of the screen, others may prefer more subdued and natural coloring, or even a monochromatic look. Personal preferences extend well beyond color to include additional image properties such as contrast and tone. They furthermore may vary with respect to semantic image content.

A natural way to express a user's personal preferences is through a set of retouched photos that they find appealing. The photographs may be self-curated from the web or taken from the collection of a favorite photographer. Such an image set gives examples of what an automatic retouching system should aim for, but few techniques are unable to take advantage of this guidance. The automatic editing tools in the literature are mostly designed to handle only a single aspect of photo retouching, such as image cropping [Yan et al. 2015], tonal adjustment [Bychkovsky et al. 2011], and color enhancement [Wang et al. 2011; Yan et al. 2014, 2016]. Moreover, the state-of-the-art techniques for these problems are all based on machine learning with training data composed of image pairs, before and after the particular retouching operation. Paired image data is generally difficult to acquire, as images prior to retouching are usually unavailable. This is especially true of photos before and after a specific retouching step.

In this paper, we present a photo retouching system that handles a wide range of post-processing operations within a unified framework, and learns how to apply these operations based on a photo collection representing a user's personal preferences. No paired image data is needed. This is accomplished through an an end-to-end learning framework in which various retouching operations are formulated as a series of resolution-independent differentiable filters that can be jointly trained within a convolutional neural network (CNN). How to determine the sequence and parameters of these filters for a given input image is learned with a deep reinforcement learning (RL) approach guided by a generative adversarial network (GAN) that models personal retouching preferences from a given photo collection.

In contrast to many neural network solutions where the functioning is hidden within a "black box", our "white box" network can reveal its sequence of editing steps for an image, which correspond to standard retouching operations and provide some understanding of the process that it took. We show both quantitatively and through user studies that the system can produce results that plausibly match the user preferences reflected within a photo collection. Examples of these automatically retouched images are displayed in Figure 1.

The technical contributions of this work are summarized as follows:

- An end-to-end model of photo post-processing with a set of differentiable filters.
- By optimizing the model using reinforcement learning, our system can generate a meaningful operation sequence that provides users with an understanding of the given artistic style, rather than just outputting a black-box result.
- Using a GAN structure, we enable learning of photo retouching without image pairs. To our knowledge, this is the first GAN that scales with image resolution and generates no distortion artifacts in the image.
- Through extensive experiments, we qualitatively and quantitatively validate our model and learning framework. We show that our method not only provides an effective end-to-end post-processing tool that aids ordinary users, but also can help advanced users to reverse-engineer the style of an automatic filter.

2 RELATED WORK

Automatic Photo Retouching. The state-of-the-art methods for automatic retouching are mainly based on supervised learning from paired images, which are obtained before and after editing by an expert photographer. Most of these methods extract handcrafted features, such as intensity distributions and scene brightness, from an input image and learn to determine editing parameters with respect to them. This approach has been employed for individual types of post-processing operations, including global tonal adjustment using Gaussian processes regression [Bychkovsky et al. 2011], image cropping using support vector machines [Fang et al. 2014; Yan et al. 2015], and color adjustment using a learning-to-rank approach [Yan et al. 2014] or with binary classification trees and parametric mapping models [Wang et al. 2011].

With recent developments in machine learning, more informative features have been extracted from images using deep convolutional neural networks. In contrast to the low-level image properties represented by handcrafted features, the features from deep learning encode high-level semantic information, from which context-dependent edits can be learned. Deep CNNs have led to clear improvements in a wide range of computer graphics applications, including 3D mesh labeling [Guo et al. 2016] and locomotion modeling [Peng et al. 2016]. For photo retouching, CNNs have been utilized for spatially varying color mapping based on semantic information together with handcrafted global and local features [Yan et al. 2016]. More recently, a CNN was trained to predict local affine transforms in bilateral space [Gharbi et al. 2017], which can serve as an approximation to edge-aware image filters and color/tone adjustments.

Our system utilizes deep CNNs as well, but differs from these previous works in that it generates a meaningful sequence of edits that can be understood and reproduced by users. Moreover, it performs learning *without* paired image data. Not only is collecting unpaired data more practical from a user's perspective, but we believe that it more closely conforms with the task of retouching. Unlike the one-to-one mapping that is implicit in supervised learning, retouching is inherently a one-to-many problem, since for a given input image there exist many possible solutions that are consistent with a retouching style. Instead of learning to convert a given photograph into a specific result, our technique learns to transform an image into a certain style as represented by a photo collection.

Related to this is a method for finding exemplar images whose color and tone style is compatible with a given input photograph [Lee et al. 2016]. This is done by semantically matching the input to clusters of images from a large dataset, and then sampling from a set of stylized images that match these clusters in chrominance and luminance distributions. Matching through these two degrees of separation may lead to exemplar images that differ drastically from the input in content and composition (*e.g.*, a landscape photo as a style exemplar for a closeup image of a child). In such cases, mapping the chrominance distribution of the exemplar to the input may produce unusual colorings. By contrast, our system models a style from a collection of photos, and transforms the input image towards this style, rather than to the statistics of a particular image.

Data-Driven Image Generation. Early methods for data-driven image generation addressed problems such as texture synthesis [Efros and Leung 2014] and super-resolution [Freeman et al. 2002] through sampling or matching of image patches from a database. To generate a certain class of images such as handwritten digits or human faces, there has been some success using variational autoencoders [Kingma and Welling 2014; Rezende et al. 2014], which construct images from a compressed latent representation learned from a set of example images.

Generative Adversarial Networks. Recently, significant progress in data-driven image generation has been achieved through generative adversarial networks [Goodfellow et al. 2014]. GANs are composed of two competing networks, namely a generator that learns to map from a latent space to a target data distribution (*e.g.*, natural images), and a discriminator that learns to distinguish between instances of the target distribution and the outputs of the generator. The generator aims to better mimic the target data distribution based on feedback from the discriminator, which likewise seeks to improve its discriminative performance. Through this adversarial process, GANs have generated images with a high level of realism [Radford et al. 2016].

A conditional variant of GANs [Mirza and Osindero 2014] makes it possible to constrain the image generation using information in an input image. Conditional GANs have been applied to image inpainting conditioned on the surrounding image context [Pathak et al. 2016], inferring photographic images from surface normal maps [Wang and Gupta 2016], super-resolution from a low-resolution input [Ledig et al. 2016], and image stylization for an input image and a texture example [Li and Wand 2016]. These image-to-image translation problems were modeled within a single framework under paired [Isola et al. 2016] and unpaired [Zhu et al. 2017] settings. These two methods are based on the observation that GANs learn a loss function that adapts to the data, so they can be applied in the same way to different image-to-image translation tasks. For CycleGAN [Zhu et al. 2017] with unpaired training data, the translations are encouraged to be "cycle consistent", where a translation by the GAN from one domain to another should be reverted back to the original input by another counterpart GAN.

Our system also uses a type of conditional GAN, but instead of directly generating an image, it outputs the parameters of filters to be applied to the input image. As the filters are designed to be content-preserving, this approach maintains the semantic content and spatial structure of the original image, as is the case for CycleGAN. Also, since the filters are resolution-independent, they can be applied to images of arbitrary size (*e.g.*, 24-megapixel photos), even though GANs in practice can generate images of only limited resolution (*e.g.*, 512×512px in CycleGAN). In addition, the generated filtering sequence represents conventional post-processing operations understandable to users, unlike the black-box solutions of most CNNs. Concurrently to our work, another deep learning based solution is proposed in [Fang and Zhang 2017], which also makes use of a conditional GAN structure, but only for the "dramatic mask" part of the system. In addition, multiple operations are learned separately, while in our work operations are optimized elegantly as a whole, guided by the RL and GAN architecture.

Reinforcement Learning. Different from existing GAN architectures, our conditional GAN is incorporated within a reinforcement learning (RL) framework to learn a sequence of filter operations. RL provides models for decision making and agent interaction with the environment, and has led to human-level performance in playing Atari games [Mnih et al. 2013] and even defeating top human competitors at the game of Go [Silver et al. 2016]. In graphics, RL has been successfully used for character animation [Peng et al. 2015, 2016, 2017; Peng and van de Panne 2017]. For natural language generation, a combination of RL and GAN was employed in [Yu et al. 2017] so that sequences consisting of discrete tokens can be effectively produced. In our work, a filtering sequence is modeled as a series of decision making problems, with an image quality evaluator defined by the GAN discriminator as the environment.

On-camera enhancement of image quality. For mobile phone cameras, methods have been developed to automatically enhance photos within the imaging pipeline. These enhancements have included image denoising [Liu et al. 2014] as well as exposure adjustment and tone mapping [Hasinoff et al. 2016]. These techniques are aimed at improving generic image quality and do not address emulating retouching styles. Also, these methods operate on a set of burst images to obtain data useful for their tasks. Our work can be employed in conjunction with such methods, for further processing to improve or personalize the photographic style.

3 THE MODEL

Given an input RAW photo, the goal of our work is to generate a result that is retouched to match a given photo collection. The photo collection may represent a specific photographic style, a particular photographer, or a more general range of image appearance, such as an assembled set of photos that the user likes. In this section, we elaborate on our modeling of the retouching process.

3.1 Motivation

In contrast to most methods that address a single post-processing operation, a more comprehensive retouching process needs to account for different types of edits and how to apply them collectively. For a human editor, retouching is done as a series of editing steps, where each step is normally decided based on the outcome of the previous step. This reliance on visual feedback exists even within a single step of the process, since the parameters for an operation, often controlled with a slider, are interactively adjusted while viewing real-time results, as shown in Figure 2.

Certainly, feedback is critical for choosing an operation and its parameters. A photographer cannot in general determine a full operation sequence from viewing only the original input image. We postulate that an automatic retouching system would also benefit from feedback and can more effectively learn how to select and apply a single operation at a time based on feedback than to infer the final output directly from the input. Moreover, modeling retouching as a sequence of standard post-processing operations helps to maintain the photorealism of the image and makes the automatic process more understandable to users.

We note that the notion of learning an operation sequence was used in a learning-to-rank model for automatic color adjustment [Yan et al. 2014]. Unlike their supervised approach which is trained on collected sequences of editing operations from expert photographers, our system requires much less supervision, needing only a set of retouched photos for training.

3.2 Post-processing as a decision-making sequence

Based on this motivation, the retouching process can naturally be modeled as a sequential decision-making problem, which is a problem commonly addressed in reinforcement learning (RL). RL is a subarea of machine learning related to how an agent should act within an environment to maximize its cumulative rewards. Here, we briefly introduce basic concepts from RL and how we formulate retouching as an RL problem.

We denote the problem as $\mathcal{P} = (\mathcal{S}, \mathcal{A})$ with $\mathcal{S}$ being the **state space** and $\mathcal{A}$ the **action space**. Specifically in our task, $\mathcal{S}$ is the space of images, which includes the RAW input image and all intermediate results in the automatic process, while $\mathcal{A}$ is the set of all filter operations. A **transition function** $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ maps input state $s \in \mathcal{S}$ to its outcome state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$. State transitions can be expressed as $s_{i+1} = p(s_i, a_i)$. Applying a sequence of filters to the input RAW image results in a **trajectory** of states and actions:

$$t = (s_0, a_0, s_1, a_1, \dots, s_{N-1}, a_{N-1}, s_N)$$

where $s_i \in \mathcal{S}$, $a_i \in \mathcal{A}$ are states and actions, N is the number of actions, and s_N is the stopping state, as shown in Figure 3. A central element of RL is the **reward function**, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which evaluates actions given the state. Our goal is to select a **policy** π that maximizes the accumulated reward during the decision-making process. For this, we use a **stochastic policy** agent, where the policy $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})$ maps the current state $s \in \mathcal{S}$ to $\mathbb{P}(\mathcal{A})$, the set of probability density functions over the actions. When an agent enters a state, it samples one action according to the probability density functions, receives the reward, and follows the transition function to the next state.

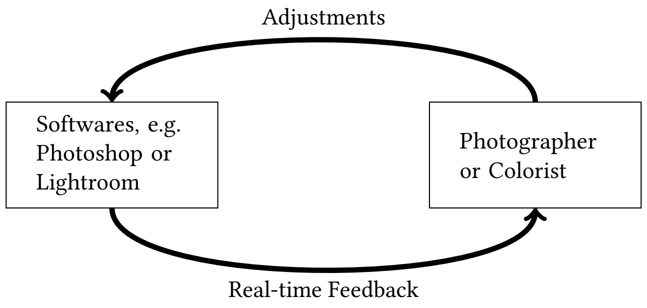


Fig. 2. Information flow in interactive photo post-processing. Our method follows this scheme by modeling retouching as a decision-making sequence.

Given a trajectory $t = (s_0, a_0, s_1, a_1, \dots, s_N)$, we define the **return** r_k^γ as the summation of discounted rewards after s_k:

$$r_k^\gamma = \sum_{k'=0}^{N-k} \gamma^{k'} r(s_{k+k'}, a_{k+k'}), \tag{1}$$

where $\gamma \in [0, 1]$ is a **discount factor** which places greater importance on rewards in the nearer future. To evaluate a policy, we define the **objective**

$$J(\pi) = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mathcal{S}_0 \\ t \sim \pi}} \left[r_0^\gamma | \pi \right], \tag{2}$$

where s_0 is the input image, and $\mathcal{S}_0$ is the input dataset. Intuitively, the objective describes the expected return over all possible trajectories induced by the policy π. The goal of the agent is to maximize the objective $J(\pi)$, which is related to the final image quality by the reward function r, as images (states) with high quality are more greatly rewarded.

The expected total discounted rewards on states and state-action pairs are defined by **state-value functions** V, and **action-value functions** Q:

$$V^\pi(s) = \mathop{\mathbb{E}}_{\substack{s_0 = s \\ t \sim \pi}} \left[r_0^\gamma \right] \tag{3}$$

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{\substack{s_0 = s \\ a_0 = a \\ t \sim \pi}} \left[r_0^\gamma \right]. \tag{4}$$

To fit our problem into this RL framework, we decompose actions into two parts: a discrete selection of filter a_1 and a continuous decision on filter parameters a_2. The policy also consists of two parts: $\pi = (\pi_1, \pi_2)$. π_1 is a function that takes a state and returns a probability distribution over filters, *i.e.* choices of a_1; and π_2 is a function that takes (s, a_1) and then directly generates a_2. Note that π_1 is stochastic and requires sampling. Since there are practical challenges in sampling a continuous random variable, we follow recent practices by treating π_2 deterministically, as described in section 5.2.

4 FILTER DESIGN

In this section, we discuss the design of filters, *i.e.* the action space $\mathcal{A}$ in our model.

4.1 Design Principles

For our system, we require the designs to adhere to the following properties.

Differentiable. For gradient-based optimization of the policy π, the filters need to be differentiable with respect to their filter parameters. This differentiability is needed to allow training of the CNN by backpropagation. Clearly, not all filters can be trivially modeled as basic neural network layers; therefore, we propose approximations

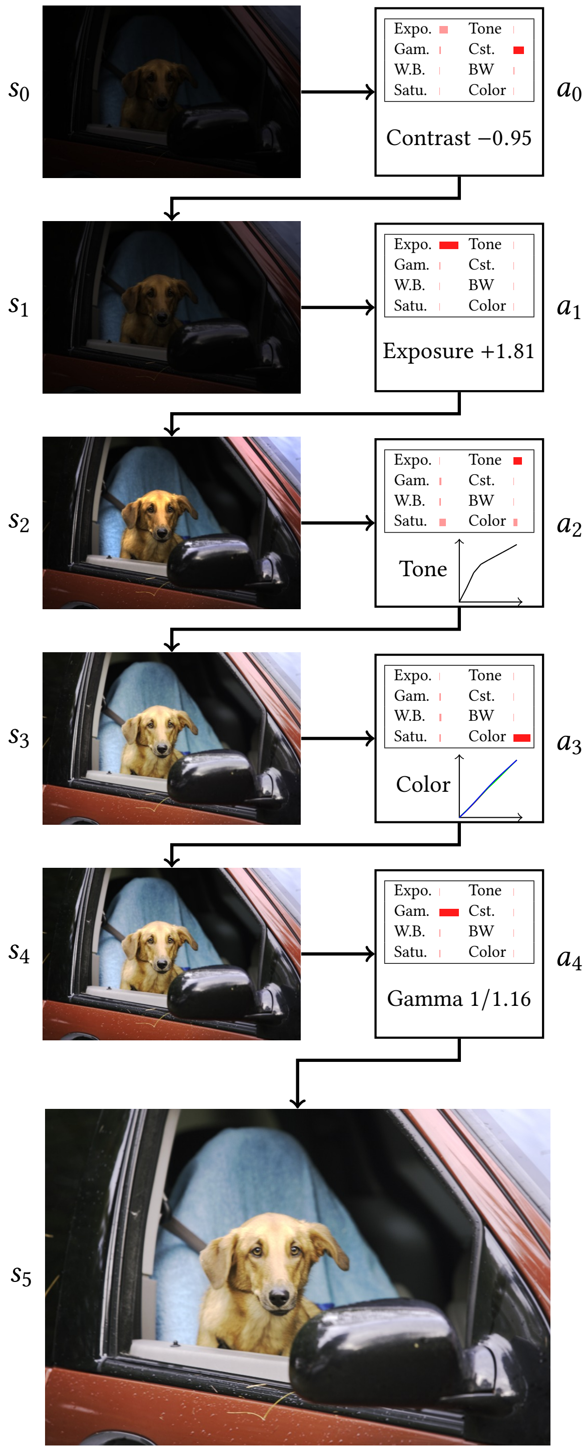


Fig. 3. An example trajectory of states (images) and actions (filter operations).

of such filters, such as piecewise linear functions in place of smooth curves, to incorporate them within our framework.

Resolution-independent. Modern digital sensors capture RAW images at a high resolution (*e.g.*, $6,000 \times 4,000$px) that is computationally impractical for CNN processing. Fortunately, most editing adjustments can be determined without examining an image at such high resolutions, thus allowing us to operate on downsampled versions of the RAW images. Specifically, we determine filter parameters on a low-resolution (64×64) version of a RAW image and then apply the same filter on the original high-resolution image. This strategy is similar to that used by Gharbi et al. [2017] to reduce computation on mobile devices. To this end, the filters need to be resolution-independent.

Note that most GAN-based image generation techniques, like CycleGAN [Zhu et al. 2017], generate images of resolution at around 512×512px, since higher resolutions lead to not only greater computational costs but also a significantly more challenging learning task that requires greater training time and training data. We experimentally compare our model to CycleGAN in section 6.1.

Understandable. The filters should represent operations that have an intuitive meaning, so that the generated operation sequence can be understood by users. This would be more interesting and instructive to users than a "black-box" result. It would also enable

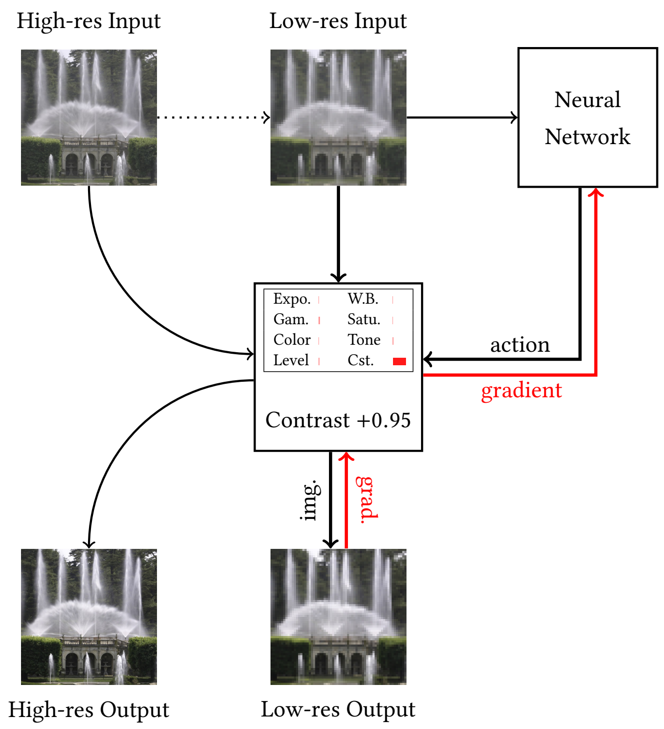


Fig. 4. The design principles of our filters (*Contrast filter* in this example): 1) They are **differentiable** and can thereby provide gradients for neural network training; 2) **Arbitrary-resolution** images can be processed with the filter; 3) What the filter does should be **understandable** by a human user.

Table 1. Simple pixel-wise filters. For more details about the filters, please refer to the supplemental material.

Operation	Parameters	Filter
Exposure	E: exposure value	$p_O = 2^E p_I$
White Balance	W_r, W_g, W_b: factors	$p_O = (W_r r_I, W_g g_I, W_b b_I)$
Color curves	$C_{i,k}$: curve param.	$p_O = (L_{C_r}(r_I), L_{C_g}(g_I), L_{C_b}(b_I))$

them to further adjust the parameters if they want. Though an alternative is to generate a black-box result and then let users apply edits to it, the black-box transformation might not be invertible, leaving users unable to undo any unwanted effects.

These three design principles are illustrated in Figure 4.

4.2 Filter Details

Based on the aforementioned design principles, we developed filters that map an input pixel value $p_I = (r_I, g_I, b_I)$ to an output pixel value $p_O = (r_O, y_O, g_O)$. Standard color and tone modifications, such as exposure change, white balancing, and color curve adjustment, can be modeled by such pixel-wise mapping functions. Examples of operations implemented in our system are listed in Table 1 and visualized in Figure 5. Color curve adjustment, i.e., a channel-independent monotonic mapping function, requires special treatment for its filter to be differentiable, as described in the following.

Curve representation. We approximate curves as monotonic piecewise-linear functions. Suppose we represent a curve using L parameters, denoted as $\{t_0, t_1, \ldots, t_{L-1}\}$. With the prefix-sum of parameters defined as $T_k = \sum_{l=0}^{k-1} t_l$, the points on the curves are represented as $(k/L, T_k/T_L)$. For this representation, an input intensity $x \in [0, 1]$ will be mapped to

$$f(x) = \frac{1}{T_L} \sum_{i=0}^{L-1} \text{clip}(L \cdot x - i, 0, 1) t_k.$$

Note that this mapping function is now represented by differentiable parameters, making the function differentiable with respect to both x and the parameters $\{t_l\}$. Since color adjustments by professionals are commonly subtle, color curves are typically close to identity. We find that eight linear segments are sufficient for modeling typical color curves.

5 LEARNING

Given the decision-making model for retouching and the differentiable filters that make optimization possible, we discuss in this section how the agent is represented by deep neural networks (DNNs), how these networks are trained, and how the reward related to the generated image quality is evaluated using adversarial learning. The whole training cycle is shown in Alg. 1, and we elaborate on the details in the following subsections.

5.1 Function approximation using DNNs

DNNs are commonly deployed as an end-to-end solution for approximating functions used in complex learning tasks with plentiful data. Since convolutional neural networks (CNN) are especially powerful in image-based understanding [Krizhevsky et al. 2012], we use CNNs in our work. Among the CNNs are two policy networks, which map the images into action probabilities π_1 (after softmax) or filter parameters π_2 (after tanh). For policies π_1 and π_2, the network parameters are denoted as θ_1 and θ_2, respectively, and we wish to optimize $\theta = (\theta_1, \theta_2)$ so that the objective $J(\pi_\theta)$ is maximized. In addition to the two policy networks, we also learn a value network and a discriminator network, which facilitate training as later described.

All of these networks share basically the same architecture illustrated in Figure 7, while having different numbers of output neurons according to what they output. For each CNN, we use four convolution layers, each with filters of size 4×4 and stride 2. Appended to this is a fully connected layer to reduce the number of outputs to 128, and then a final fully connected layer that further regresses the features into parameters we need from each network. The deterministic policy networks (one for each filter) for filter parameter estimation share the convolutional layers, so that the computation is made more efficient. CNNs are largely tailored for hierarchical visual recognition, and we found that naively using them results in unsatisfactory learning of agent policies and global statistics. Therefore, following [Silver et al. 2016], we concatenate extra (spatially constant) feature planes as additional color channels in the input. For the discriminator network, the additional feature planes are for the average luminance, contrast and saturation of the entire image; for the policy and value networks, the feature planes are eight boolean (zero or one) values that indicate which filters have been used, and another plane denotes the number of steps that have been taken so far in the retouching process.

5.2 Policy network training

The policy networks are trained using policy gradient methods, which employ gradient descent to optimize parameterized policies with respect to the expected return. As the policy π consists of two parts (π_1, π_2) corresponding to the two decision-making steps (*i.e.*, filter and parameter selection), they are learned in an interleaved manner.

For filter selection, we sample π_1, which is a discrete probability distribution function $\pi_1(F_k) = \mathbb{P}[a_1 = F_k]$ over all choices of filters $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$. Unlike other common differentiable operations including convolution or activation, the partial derivative $\partial J(\pi)/\partial \pi(F_k)$ cannot be directly calculated, which presents a challenge for backpropagation. We address this by applying the policy gradient theorem [Sutton et al. 2000] to obtain an unbiased Monte Carlo estimate of the gradient of $J(\pi)$ with respect to π_1. For filter parameter selection, policy π_2 is deterministic, so that it is easier to optimize in a continuous space, and we formulate its gradient using the deterministic policy gradient theorem [Silver et al. 2014]. The policy gradients are thus expressed as

$$\nabla_{\theta_1} J(\pi_\theta) = \mathop{\mathbb{E}}_{\substack{s \sim \rho^\pi \\ a_1 \sim \pi_1(s) \\ a_2 = \pi_2(s, a_1)}} [\nabla_{\theta_1} \log \pi_1(a_1|s) Q(s, (a_1, a_2))], \quad (5)$$

$$\nabla_{\theta_2} J(\pi_\theta) = \mathop{\mathbb{E}}_{\substack{s \sim \rho^\pi \\ a_2 = \pi_2(s, a_1)}} [\nabla_{a_2} Q(s, (a_1, a_2)) \nabla_{\theta_2} \pi_2(s, a_1)], \quad (6)$$

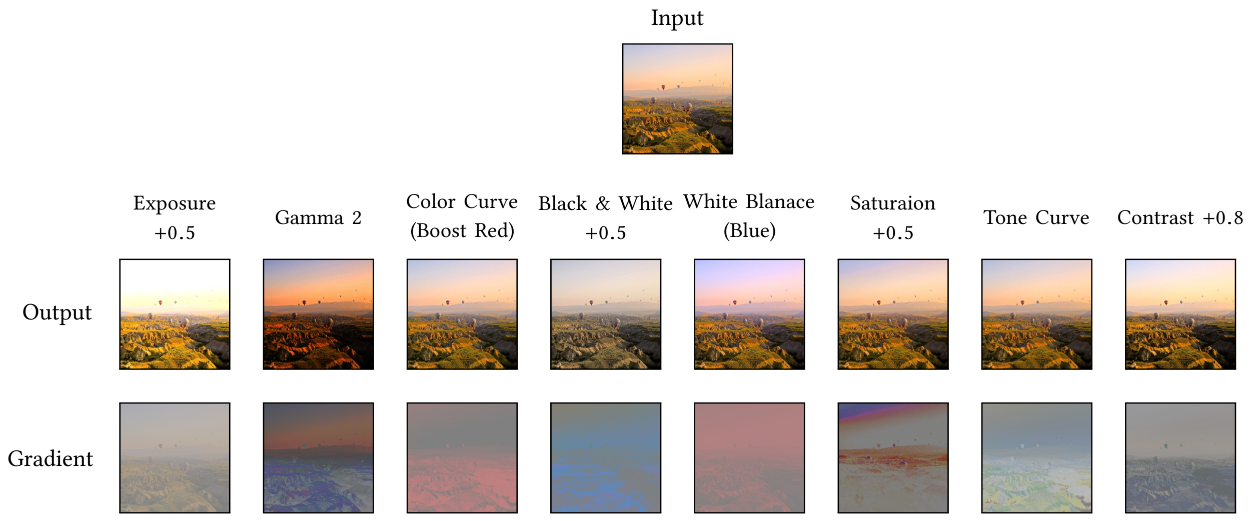


Fig. 5. Visualizations of the eight differentiable filters. Gradients are displayed with a +0.5 offset so that negative values can be properly viewed. For the white balance filter, we visualize the gradient with respect to the red channel parameter; for the tone/color curves, we differentiate with respect to the first parameter of the curve/red curve.

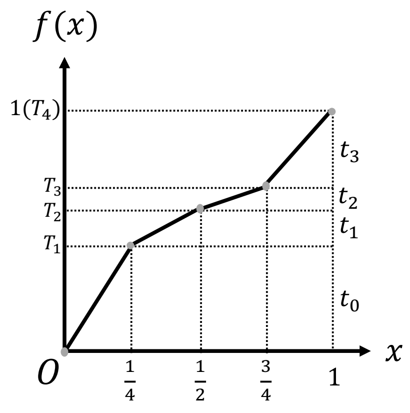


Fig. 6. Representation of a curve using regressed parameters.

where ρ^π is the discounted state distribution defined as

$$\rho^\pi(s) = \sum_{k=0}^{\infty} \mathbb{P}(s_k = s)\gamma^k,$$

and Q is the value function defined in Eqn. 4.

To calculate these gradients, we apply the actor-critic algorithm [Sutton et al. 2000], where the actor is represented by the policy networks and the critic is the value network, which learns the state-value function $V^\nu : \mathcal{S} \to \mathbb{R}$ of Eqn. 3 for estimating expected returns. With the critic, the action-value function Q^π can be computed by unfolding its definition (Eqn. 4) and expressing it in terms of the state-value function:

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\substack{s_0=s\\a_0=a\\t\sim\pi}} [r(s_0,a_0) + \gamma V^\pi(p(s_0,a_0))].$$

Plugging this into Eqn. 6 gives us the supervision signal for learning π_2.

We optimize the value network by minimizing

$$L_v = \mathop{\mathbb{E}}_{s\sim\rho^\pi, a\sim\pi(s)} \left[\frac{1}{2}\delta^2\right],$$

where δ is the temporal difference (TD) error: $\delta = r(s,a)+\gamma V(p(s,a)) - V(s)$. Note that δ represents the Monte Carlo estimate of the **advantage** $A(s,a) = Q(s,a) - V(s)$, i.e., how much the value of action a exceeds the expected value of actions at state s. For calculating the gradient of π_1 in Eqn. 5, the Q-value $Q(s,a)$ can be substituted by the advantage $A(s,a)$, which effectively reduces sample variance and can conveniently be computed as the TD error δ. Note that the gradient of π_2 requires no Monte Carlo estimation, thus we directly calculate it by applying the chain rule on the gradient of Q, instead of using the advantage A.

Reward and discount factor. The ultimate goal is to obtain the best *final* result after all operations. For this, we set the reward as the incremental improvement in the quality score (modeled by a discriminator network in the following subsection) plus penalty terms (described in Sec. 5.4). We set the discount factor as $\gamma = 1$ and allow the agent to make five edits to the input image. This number of edits was chosen to balance expressiveness and succinctness of the operation sequence.

5.3 Quality evaluation via adversarial learning

To generate results as close to the target dataset as possible, we employ a GAN, which is composed of two parts, namely a generator (i.e., the actor of the previous subsection in our case) and a discriminator. The two parts are optimized in an adversarial manner: the discriminator is trained to tell if the image is from the target dataset or was generated by the generator; the actor aims to “fool” the discriminator by generating results as close to the target dataset

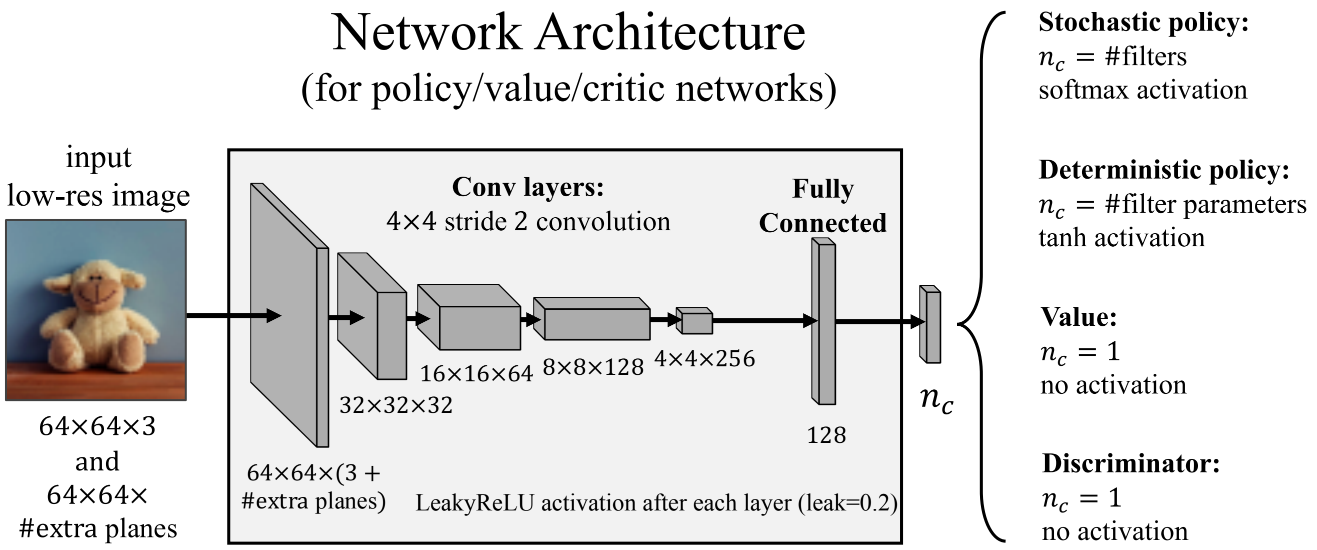


Fig. 7. The general network structure shared by all networks in our system.

ALGORITHM 1: Training procedure

```
Input: Input datasets $D_{\text{RAW}}$ and $D_{\text{retouched}}$; batch size $b = 64$, learning
    rates $\alpha_\theta = \alpha_\omega = 5 \times 10^{-5}$, $\alpha_v = 5 \times 10^{-4}$, $n_{\text{critic}} = 5$
Output: Actor model $\theta = (\theta_1, \theta_2)$, critic model $v$, and discriminator model
    $w$
Initialize replay memory with 2, 048 RAW images;
while $\theta$ has not converged do
    for $i$ in $1..n_{critic}$ do
        Sample a batch of $b$ finished images from replay memory;
        Sample a batch of $b$ target images from $\mathcal{D}_{\text{target}}$;
        $w \leftarrow w - \alpha_w \nabla_w L_w$;
    end
    Draw a batch $B$ of $b$ images from replay memory;
    Delete images in the batch that are already finished;
    Refill deleted images in the batch using those from $D_{\text{RAW}}$;
    Apply one step of operation to the images: $B' = \text{Actor}(B)$;
    $\theta_1 \leftarrow \theta_1 + \alpha_\theta \nabla_{\theta_1} J(\pi_\theta)$;
    $\theta_2 \leftarrow \theta_2 + \alpha_\theta \nabla_{\theta_2} J(\pi_\theta)$;
    $v \leftarrow v - \alpha_v \nabla_v L_v$;
    Put new images $B'$ back into replay memory;
end
```

as possible, so that the discriminator cannot distinguish the difference. The two networks are trained simultaneously, and an ideal equilibrium is achieved when the generated images are close to the targets.

In this work, we use a popular variant of the traditional GAN called the Wasserstein GAN (WGAN) [Arjovsky et al. 2017], which uses the Earth Mover's Distance (EMD) to measure the difference between two probability distributions. It has been shown to stabilize GAN training and avoid vanishing gradients. The loss for the discriminator[1] D is defined as

$$L_w = \mathop{\mathbb{E}}_{s \in \rho^\pi} [D(s)] - \mathop{\mathbb{E}}_{s \in \text{target dataset}} [D(s)]\,. \tag{7}$$

The discriminator D is modeled as a CNN with parameters denoted as w. The "negative loss" (quality score) for the generator, whose increment serves as a component of the reward in our system, is

$$-L_{\text{actor}} = \mathbb{E}\,[D(s)]\,. \tag{8}$$

Intuitively, the discriminator aims to give large values for images in the target collection, and small ones to retouched images produced by the generator. On the other hand, the actor (generator) aims to submit an output at a state where the discriminator gives a larger value, meaning that the final image appears more similar to those in the target dataset. Following [Gulrajani et al. 2017], we use a gradient penalty instead of weight clipping in the discriminator.

5.4 Training stabilization

Both RL algorithms and GANs are known to be hard to train. To address this issue, we utilized the following strategies to stabilize the training process.

Exploitation vs. exploration. A well-known tradeoff exists between exploitation and exploration, namely whether to devote more attention on improving the current policy or to try a new action in search of potentially greater future reward. This is especially challenging for our two-stage decision making problem, as focusing on one filter may lead to under-exploitation of filter parameter learning for other filters. To avoid such local minima, we penalize π_1 if its action

[1]The discriminator is referred to as the "critic" in [Arjovsky et al. 2017]. We use the term "discriminator" here to distinguish it from the critic in our actor-critic framework.

proposal distribution is too concentrated, i.e., has low entropy. This is done by reducing its reward:

$$R' = R - 0.05\left(\log|\mathcal{F}| + \sum_{F\in\mathcal{F}} \pi_1(F)\log\pi_1(F)\right).$$

In addition, we found that the agent may use a filter repeatedly during a retouching process, such as by applying two consecutive exposure adjustments rather than combining them into a single step. For a more concise retouching solution, we "teach" the agent to avoid actions like this by penalizing filter reuse: if the agent uses a filter twice, the second usage will incur an additional penalty of -1. To implement this, the agent needs to know what filters have been applied earlier in the process, so we append this usage information as additional channels in the image (denoted as *state planes* in Fig. 7). Encouraging the agent to exploit each filter to its maximum potential also leads to greater exploration of different filters.

Experience Replay. Trajectories at consecutive time steps in the training process are highly correlated. Since such temporally unvaried training data can drive the network to a local minimum, we improve training stability through the use of *experience replay* [Lin 1993], where previously processed states are randomly sampled from the replay memory of $2,048$ images and used in training. This strategy helps to reduce sample correlation and stabilizes training. In our adversarial learning setting, this approach also helps to reduce model oscillation as observed in [Shrivastava et al. 2016].

6 RESULTS

In this section, we present implementation details, a validation, and applications of our system.

Implementation details. TensorFlow [Abadi et al. 2015], a deep learning framework which provides automatic differentiation, is used to implement our system. Following the design of filters presented in Section 4, the retouching steps are represented as basic differentiable arithmetic operations. For estimation of retouching actions and parameters, we downsample the high-resolution input image to $64 \times 64px$. The estimated actions and parameters are subsequently applied to the full-resolution image at run time.

All of the networks are optimized using Adam [Kingma and Ba 2015], with a base learning rate of 5×10^{-5} for both of the policy networks and the discriminator, and 5×10^{-4} is used for the value network. During training, the base learning rate is exponentially decayed to 10^{-3} of the original value. Training takes less than 3 hours for all experiments.

Efficiency and model size. Thanks to the resolution-independent filter design, the computation of our method is fast: an unoptimized version takes $30ms$ for inference on an NVIDIA TITAN X (Maxwell) GPU. The model size is small ($< 30MB$) and therefore can be conveniently shipped with a mobile application or digital camera. This opens up the possibility of providing users with automatically retouched images in the camera viewfinder in real time.

Datasets. We utilize two sources of training data:

- The MIT-Adobe FiveK Dataset. Bychkovsky et al. [2011] compiled a photo dataset consisting of $5,000$ RAW images and retouched versions of each by five experts. In this work, we randomly separate the dataset into three parts: **(part 1)** $2,000$ input RAW images, **(part 2)** $2,000$ retouched images by retoucher C, and **(part 3)** $1,000$ input RAW images for testing. The three parts have no intersection with each other.
- The 500px Dataset. We crawled professionally retouched photos from two artists on 500px.com. The two sets of data have relatively consistent styles, and are comprised of 369 and 397 photos each.

Error Metrics. The novel learning framework enables our system to take advantage of unpaired training data, which requires error metrics different from previous work.

It is shown in [Hwang et al. 2012] and that the l_2 loss may not accurately reflect visual quality. This problem is especially apparent when a dataset exhibits multi-modal appearance or style, such as black-and-white apples retouched into red (e.g., $RGB = (1, 0, 0)$) or green ($RGB = (0, 1, 0)$) with a 50% probability for each. As pointed out in [Isola et al. 2016; Pathak et al. 2016; Zhang et al. 2016], use of simple loss functions like l_1 or l_2 can lead to "blurry" results for image generation. For the apples example, a CNN with an l_2 loss will end up generating yellow ($RGB = (0.5, 0.5, 0)$) apples, which minimizes the loss but may produce styles that do not even exist in the dataset. Multi-modality naturally exists in retouching, since the same artist may retouch an image in different ways. The inconsistent nature of retouching is exhibited in the MIT-Adobe FiveK dataset and was also observed in [Yan et al. 2016].

Therefore, even if input-output pairs do exist for some tasks, the l_2 loss may not be a suitable learning metric. However, how to automatically evaluate the perceptual quality of style learning remains an open problem. Though such metrics are difficult to design and are sometimes unreliable, for development and debugging purposes, it would still be good to have an automatic way to roughly measure how well the model fits the target data. Toward this end, we evaluate the similarity of generated images to target images based on their *distributions* of image properties. In [Isola et al. 2016], the distances of L, a, b distributions are measured using the intersections of their histograms in the output and target datasets. In our work, we use **luminance**, **contrast**, **saturation** as three descriptive features of image styles, and measure the distance of their distributions in the output and target images using histogram intersections. A detailed explanation of this metric is given in the supplemental document.

In addition to histogram intersections, we employ user studies via Amazon Mechanical Turk (AMT) for perceptual evaluation of this work. For each group of outputs from a given method, we randomly choose 100 images and ask users to rate the image. The user is presented with one output image (with target style image thumbnails, if necessary) at a time and is prompted to give a score from 1 (worst) to 5 (best) to each image, based on image quality and style. 5 ratings are collected for each image, resulting in 500 ratings for each group of outputs. Please refer to the supplemental document for more details about our AMT experiments.

6.1 End-to-end Post-Processing and Style Learning

Most previous methods for automatic photo post-processing are based on supervised learning which requires paired data [Bychkovsky et al. 2011; Dale et al. 2009; Gharbi et al. 2017; Hwang et al. 2012; Yan et al. 2016]. It is only recently that a series of works [Kim et al. 2017; Liu and Tuzel 2016; Zhu et al. 2017] based on GANs have made possible the utilization of unpaired data. We compare our results with those of CycleGAN, another deep learning approach for image generation using only unpaired data. Note that in contrast to CycleGAN, our method has no limitation on resolution, since the filters are resolution-independent and filter operations estimated from low-res images can be identically applied to high-res inputs.

We conducted three sets of experiments using RAW images from **part 1** of the MIT-Adobe FiveK dataset as input. For the target datasets we use images from expert C in the MIT-Adobe FiveK and the two artists from 500px.com, respectively. Though Pix2pix [Isola et al. 2016] needs *paired* data to work, we still include its performance on a test using paired data from **part 1** of the MIT-Adobe FiveK dataset.

For the first experiment with images from expert C as target images, visual results are shown in Figure 8 and Figure 9, and quantitative results are listed in Table 2. It can be seen that Pix2pix and CycleGAN generate vivid color but lead to edge distortions and degraded image quality, making them unsuitable for high-quality post-processing tasks. Using the publicly available implementation of CycleGAN from the authors [Zhu et al. 2017], training takes 30 hours for generating images of resolution $500 \times 333\text{px}^2$.

For the style learning experiments with the *500px* artists, quantitative results are shown in Table 3 and Table 4, and visual results are displayed in Figure 10. No comparison results can be generated for Pix2pix, since no paired training data is generally available for images downloaded from the web.

Discussion. It can be seen that our method outperforms strong baselines in the user study, with higher user ratings than Pix2pix (which relies on much stronger supervision from paired training data), likely due to the fact that our images have no blurry artifacts. CycleGAN, the unpaired version of Pix2pix, does not perform as well, likely due to much weaker supervision from only unpaired data. It is worth noting that during the user study, the image resolution we used was around 500×333px, so that the resolution problem of CycleGAN and Pix2pix may not be very pronounced. However, at higher output resolutions, it becomes clear that our method generates images of much higher quality, as shown in Figure 9. The deconvolution structure of CycleGAN and Pix2pix enable them to generate structural transformations of images, e.g. painting stripes on horses to generate zebras. However, on our task, such a capability can bring distortion artifacts. The **Contrast** histogram intersection score for CycleGAN on the artist A experiment (Table 3) is lower than the other metrics. We hypothesize the reason to be that its small receptive field (1/3 of the whole image width) does not adequately capture low-frequency image variations, which is a feature of this artist. A larger receptive field or downsampled image could be used for CycleGAN, but this would require more training data and would produce even lower-resolution outputs.

In conclusion, the results of our system on the retouching problem are very promising. We note though that Pix2pix and CycleGAN can produce extraordinary results on image translation with structural transformations, while our system is tailored for photo post-processing and is not capable of such structural transformations.

[2] We used *fineSize*=128 in the authors' implementation (https://github.com/junyanz/CycleGAN).

Table 2. Quantitative results on general post-processing (using expert C in MIT-Adobe FiveK as training target dataset).

Approach	**Histogram Intersection**			**User Rating**
	Luminance	Contrast	Saturation	
Ours	71.3%	83.7%	69.7%	3.43 ± 0.04
CycleGAN	61.4%	71.1%	82.6%	2.47 ± 0.04
Pix2pix	92.4%	83.3%	86.5%	3.37 ± 0.04
Human	-	-	-	3.30 ± 0.04
Expert C	100%	100%	100%	3.66 ± 0.03

Table 3. Quantitative results on style learning (using artist A in 500px as training target dataset).

Approach	**Histogram Intersection**			**User Rating**
	Luminance	Contrast	Saturation	
Ours	82.4%	80.0%	71.5%	3.39 ± 0.04
CycleGAN	63.6%	45.2%	71.8%	2.69 ± 0.04
500px artist A	100%	100%	100%	3.72 ± 0.04

Table 4. Quantitative results on style learning (using artist B in 500px as training target dataset).

Approach	**Histogram Intersection**			**User Rating**
	Luminance	Contrast	Saturation	
Ours	85.2%	91.7%	83.5%	3.22 ± 0.04
CycleGAN	60.1%	79.4%	83.4%	2.86 ± 0.04
500px artist B	100%	100%	100%	3.40 ± 0.04

6.2 Reverse Engineering Black-box Filters

Our work is not the first attempt to mimic the effects of black-box filters. Previous methods [Gharbi et al. 2017; Yan et al. 2016] have shown excellent results in doing so for Instagram/Photoshop filters. However, these learned filters do not reveal how the original filter works, i.e. we are only getting another black-box out of an existing one.

Our method not only generates visually pleasing results, but also reveals how this process is done step by step, as shown in Figure 1 and 3 (on expert C from the MIT-Adobe FiveK dataset), Figure 11 (on artist A from 500px), Figure 12 (on artist B from 500px) and Figure 13 (on the black-box filter "Nashville" from Instagram). This is the first time such understandable result can be obtained to the best of our knowledge.

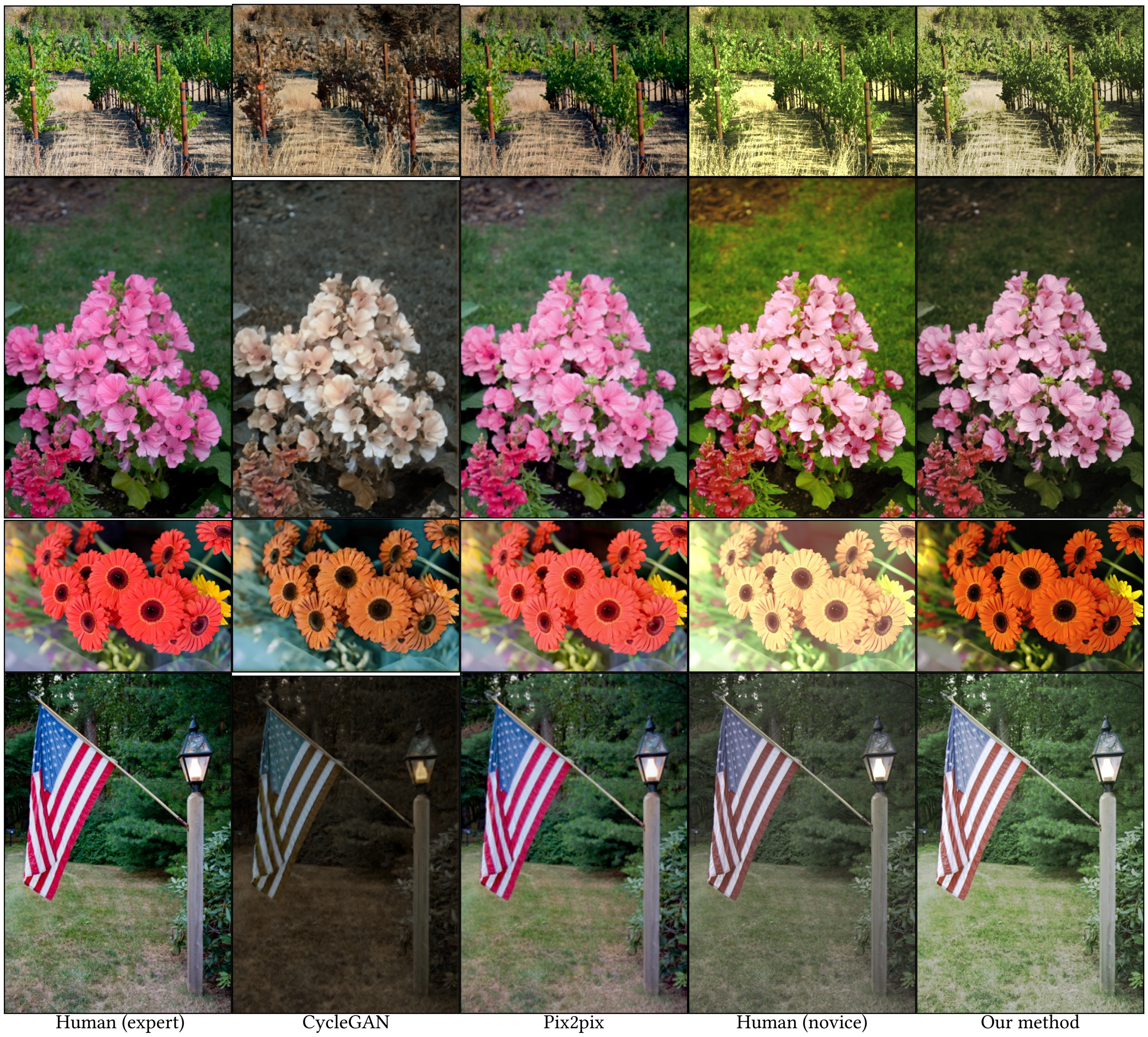


Fig. 8. Results of human expert, CycleGAN, Pix2pix, normal human, and our method on expert C from the MIT-Adobe FiveK Dataset.

Interestingly, with the help of our system, we can even write explicit code for a black-box filter based on the estimated operation sequence, as illustrated in Figure 13 and 14. We believe this capability can greatly help advanced users to gain insight into the artistic styles of particular photographers.

The variation of learned operation sequences from certain target dataset reveals how consistent the images styles are. We find that for the "Nashville" filter, the operation sequences are basically the same, while for human artists the sequences vary more. This observation matches the previous discussions regarding the error metric and the multi-modal nature of human retouching.

6.3 Comparison with human users

Unlike image classification, retouching is a challenging task for most ordinary people. While they can judge how good-looking an image is, it is often challenging for them to generate a nicely

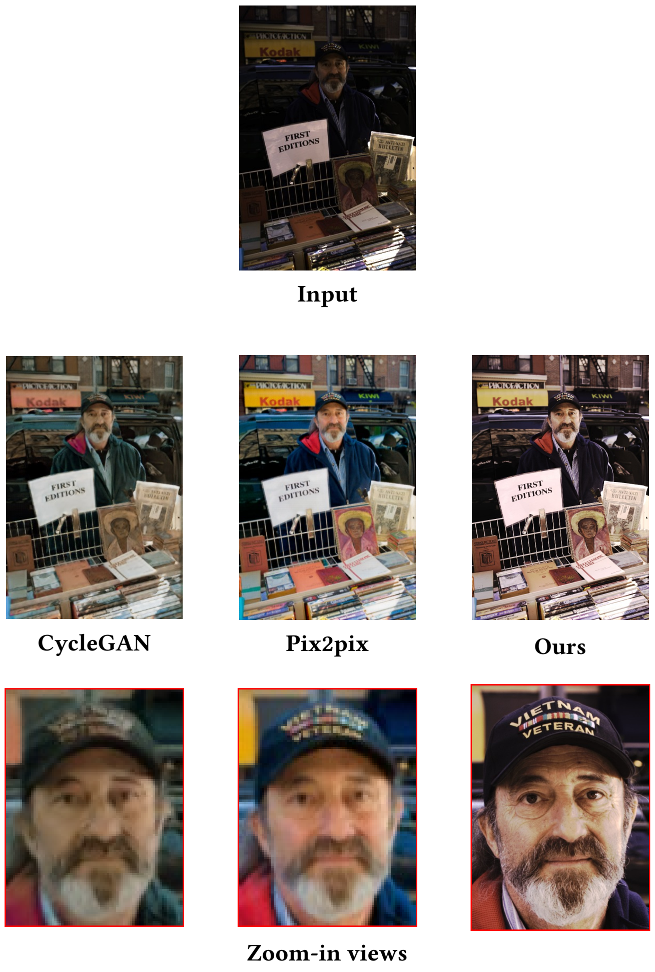


Fig. 9. Comparison to CycleGAN and Pix2pix results. While they can produce good tone and color compared to the input, our method additionally has no distortion and no limit on image resolution.

retouched photo. Though experts prefer manually retouching photos for maximum control, one of the main goals of our system is to help ordinary users to obtain better photos automatically. Therefore, we examine how normal users perform at this task, and how our system compares to them.

We developed a graphical user interface (Figure 15) to measure human performance on this task. We provide exactly the same set of operations to the user as to the network, except for curve-based edits we provide 3 control points instead of 8 to make the interface more user-friendly. To introduce our software to the user, we show them a short tutorial video before they start. 100 images from 10 users are collected, and their performance is given in Table 2. It can be seen from the user study that our method can generate results that are preferable to those produced by these users.

7 CONCLUDING REMARKS

Inspired by the retouching process of expert photographers, we proposed a general framework for automatic photo post-processing that utilizes reinforcement learning to reveal an understandable solution composed of common image manipulations, generative adversarial networks that allow training from unpaired image data, and differentiable, resolution-independent filters to make network optimization possible over a variety of editing operators and for arbitrary image sizes. The effectiveness of this method was demonstrated through quantitative and qualitative comparisons. This framework is general enough to incorporate a broader set of operations, which we hope can make it even more versatile.

Certain low-level image filters, such as for pixel-level denoising, may be challenging to model as resolution-independent, differentiable filters, and thus may not fit into our framework. Without denoising, the image noise in shadows may become more pronounced after operations that boost brightness, as seen in Figure 12. Denoising ideally should be applied to the input image prior to using our framework. Other failure cases are presented in Figure 16.

For learning to retouch photos, we have only 2×10^3 training images compared with the 1.4×10^7 images in ImageNet for image classification. It would be meaningful in future work to 1) build larger datasets of RAW photos, and 2) transfer or reuse the knowledge distilled from ImageNet to the retouching problem.

In addition, it is possible to replace the actor-critic RL architecture and the Wasserstein GAN structure with other related alternatives. We find that much human labor and expertise is required to properly set the hyper-parameters to stabilize the training process. We believe that using more stable RL and GAN components will make this process easier and lead to even better results.

Figure. 17 and 18 exhibit some final examples of retouched photos by our system. We hope that not only machines but also all interested people can understand the secrets of digital photography better, with the help of our "Exposure" system.

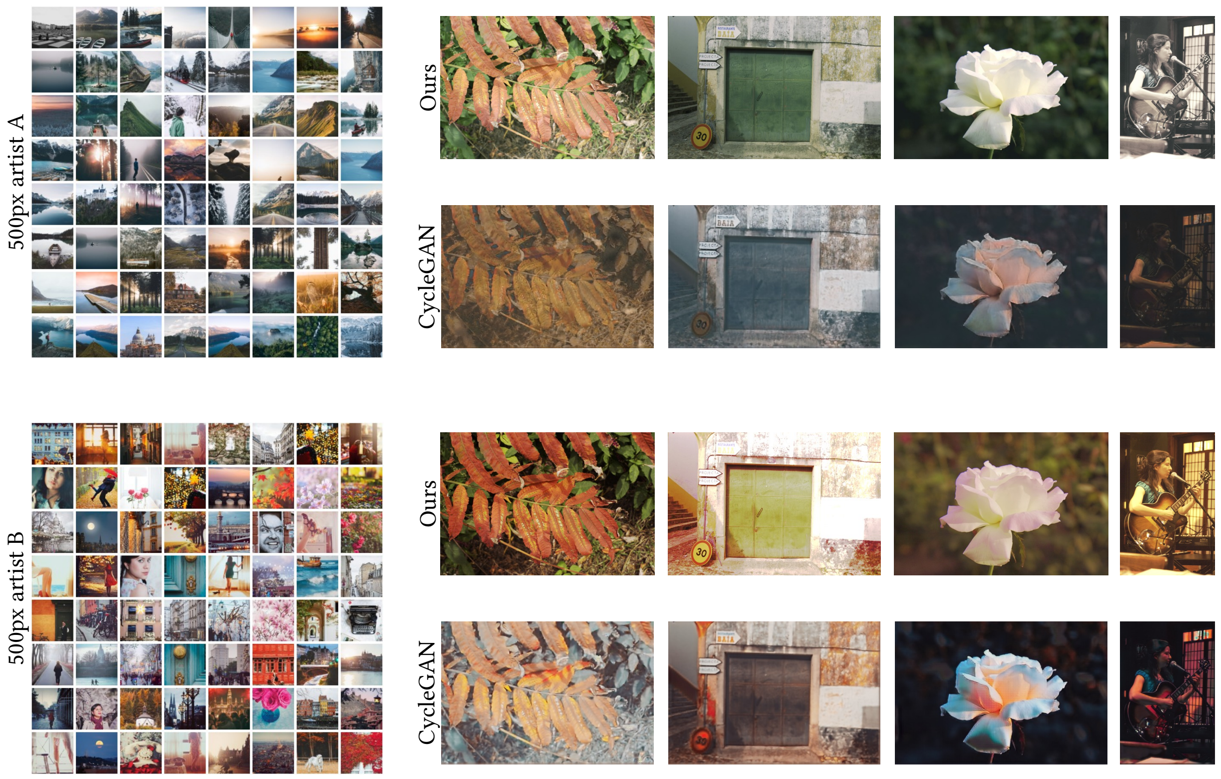


Fig. 10. Learning the styles of two artists from 500px.com, using our system and CycleGAN.

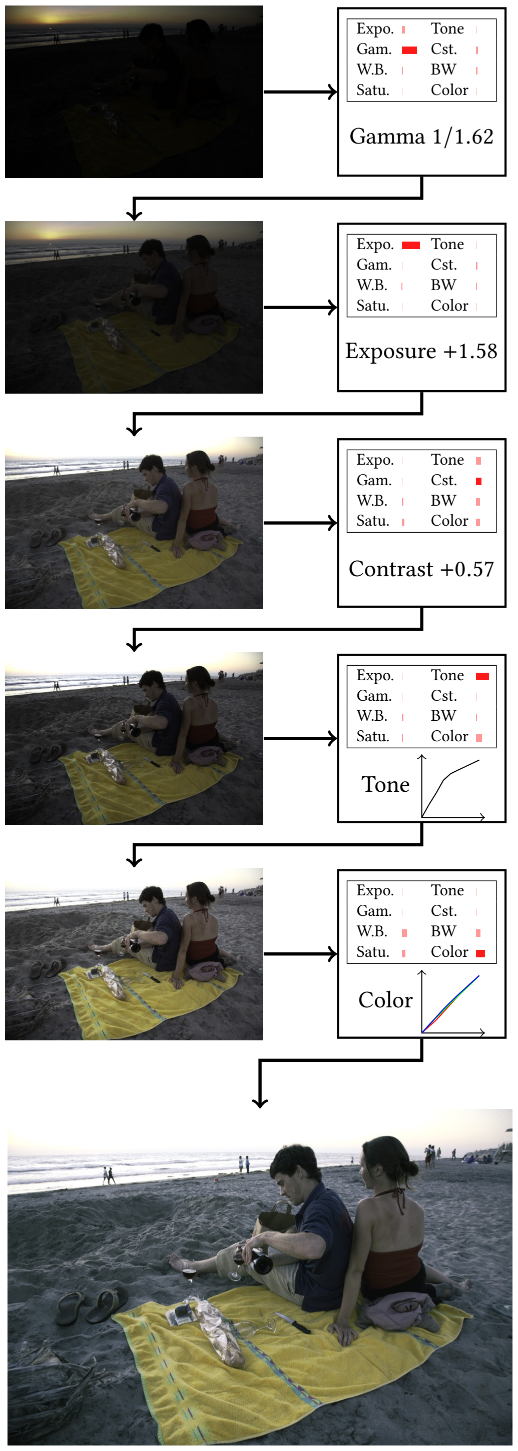


Fig. 11. Example of a learned retouching operation sequence from artist A (500px).

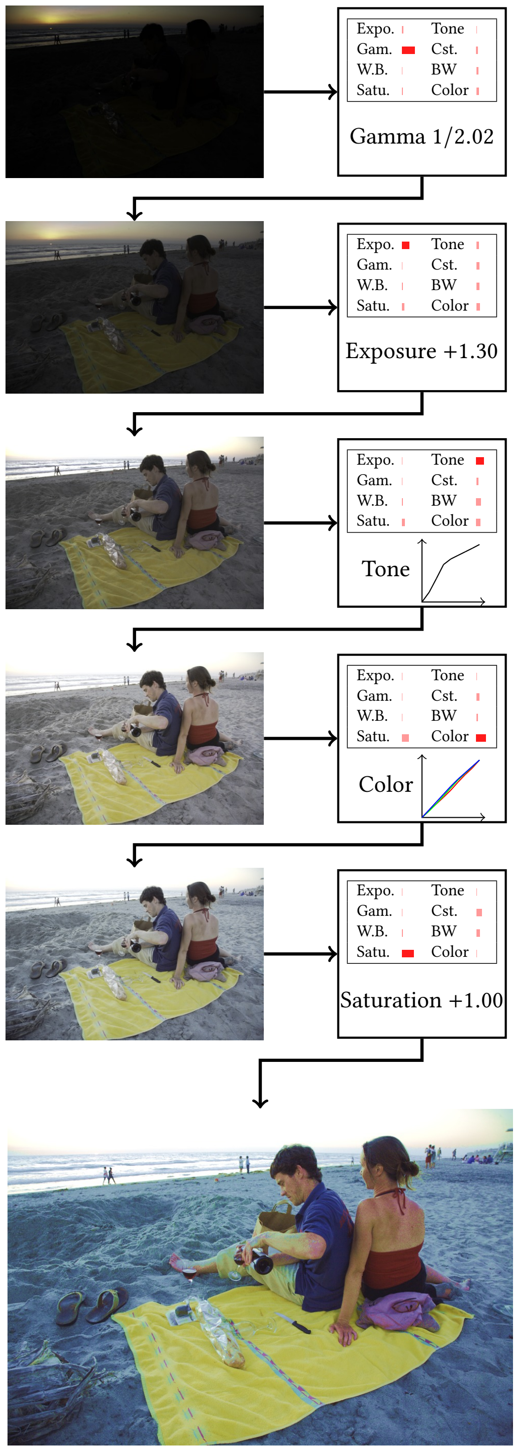


Fig. 12. Example of a learned retouching operation sequence from artist B (500px). Note that different from artist A, photos from artist B are more saturated, which is reflected in this learned operation sequence.

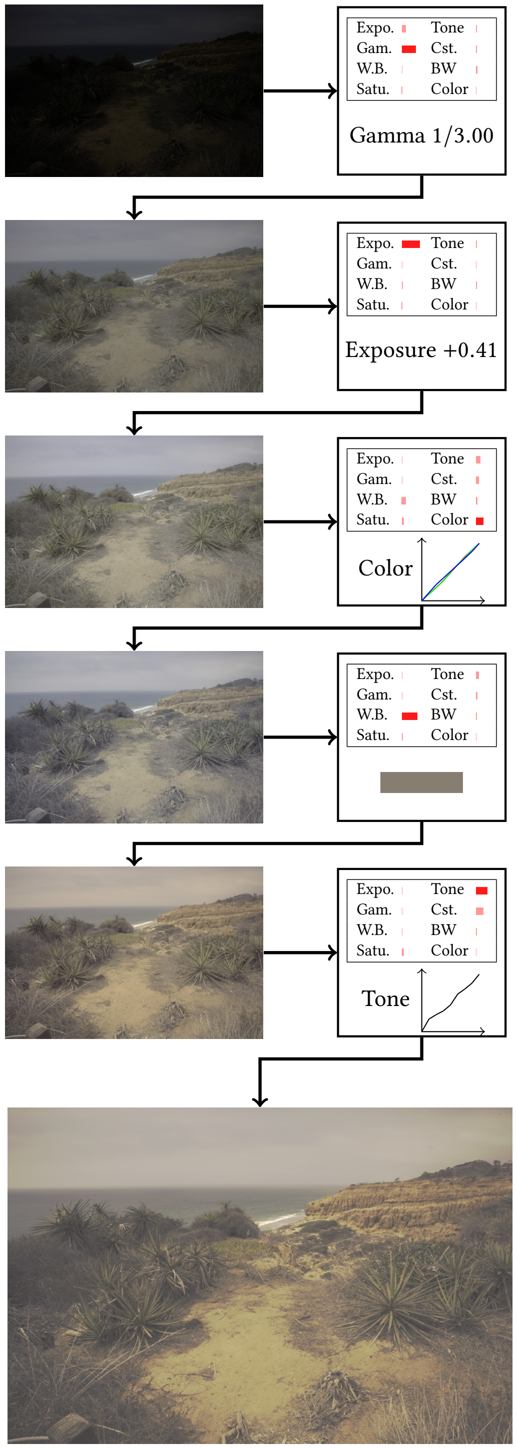
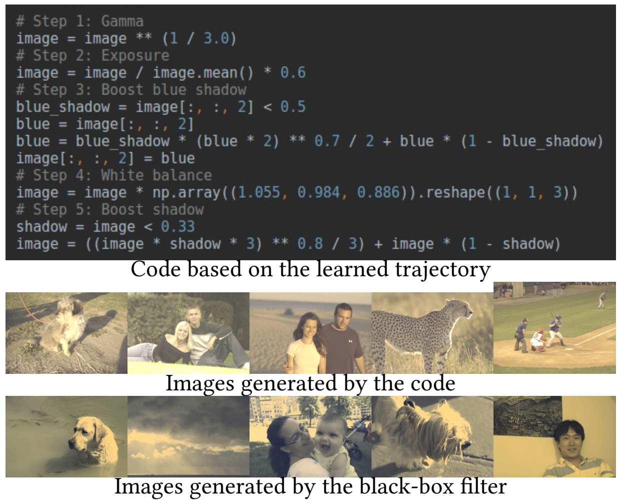


Fig. 13. Example of a learned operation sequence on the "Nashville" filter from Instagram.



```
# Step 1: Gamma
image = image ** (1 / 3.0)
# Step 2: Exposure
image = image / image.mean() * 0.6
# Step 3: Boost blue shadow
blue_shadow = image[:, :, 2] < 0.5
blue = image[:, :, 2]
blue = blue_shadow * (blue * 2) ** 0.7 / 2 + blue * (1 - blue_shadow)
image[:, :, 2] = blue
# Step 4: White balance
image = image * np.array((1.055, 0.984, 0.886)).reshape((1, 1, 3))
# Step 5: Boost shadow
shadow = image < 0.33
image = ((image * shadow * 3) ** 0.8 / 3) + image * (1 - shadow)
```

Code based on the learned trajectory
Images generated by the code
Images generated by the black-box filter

Fig. 14. With the operation sequence estimated by our system, we can write code that mimics certain black-box filters.

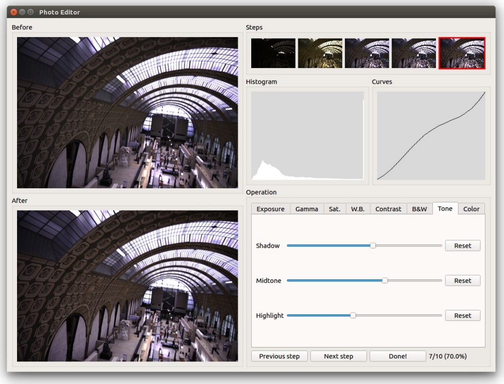


Fig. 15. The graphical user interface for collecting retouching data from ordinary users. An intensity histogram of the current image and curves of color/tone curve operations are displayed.

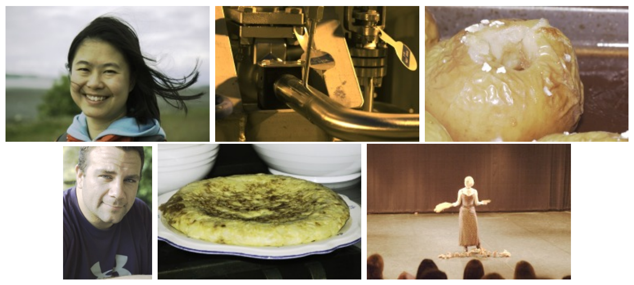


Fig. 16. Example failure cases. Our method sometimes does not produce good tones for faces, as no special consideration is taken in our general framework of this particularly important aspect of photos. Also, our system may have limited ability to improve input photos that contain poor content, composition or lighting conditions.

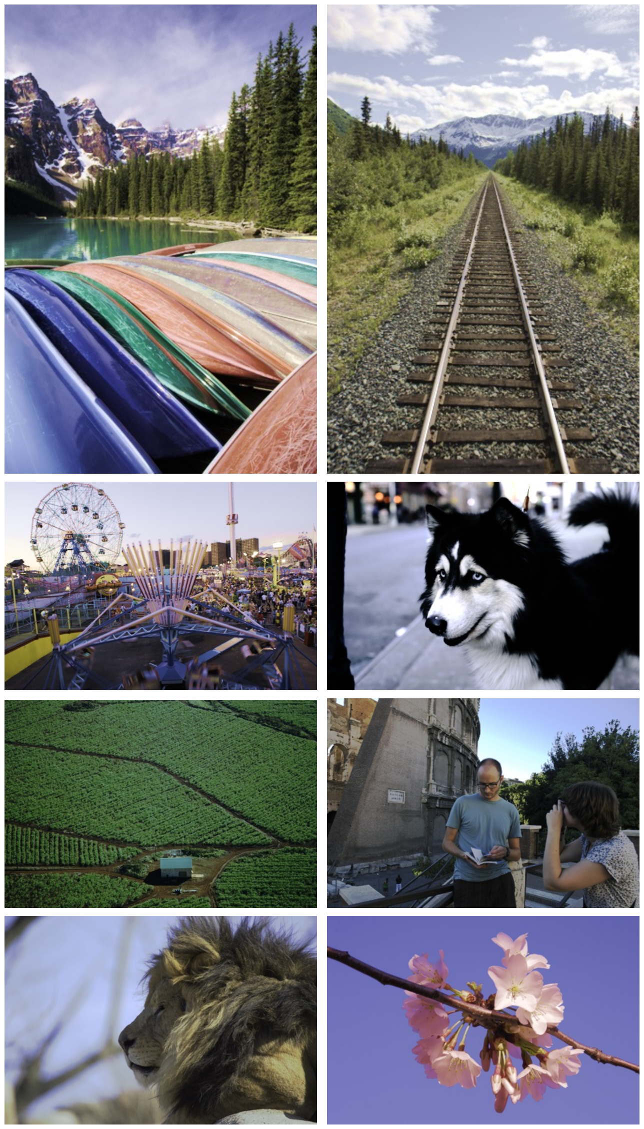


Fig. 17. More photos retouched by our system, trained on expert C from MIT-Adobe FiveK dataset.

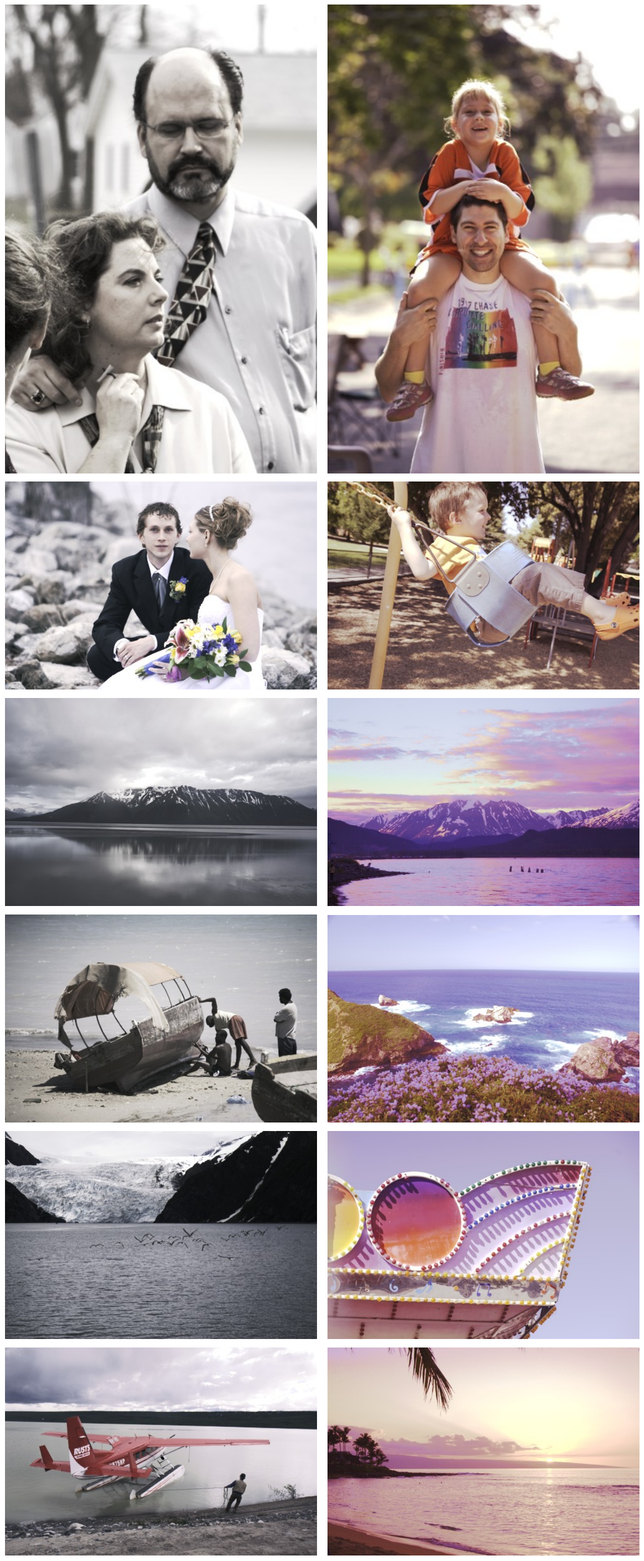


Fig. 18. More photos retouched by our system, trained on artist A (left column) and artist B (right column) from *500px*.

SUPPLEMENTAL DOCUMENT

Filter Design Details

Contrast, Saturation, and Black & White Filters. These filters are designed similarly, with an input parameter that sets the linear interpolation between the original image and the fully enhanced image, i.e.,

$$p_O = (1 - p) \cdot p_I + p \cdot \text{Enhanced}(p_I).$$

For **Contrast**:

$$\text{EnhancedLum}(p_I) = \frac{1}{2}(1 - \cos(\pi \times (\text{Lum}(p_I)))),$$

$$\text{Enhanced}(p_I) = p_I \times \frac{\text{EnhancedLum}(p_I)}{\text{Lum}(p_I)},$$

where the luminance function Lum(p) = $0.27p_r + 0.67p_g + 0.06p_b$.

For **Saturation**:

$$\text{EnhancedS}(s, v) = s + (1 - s) \times (0.5 - |0.5 - v|) \times 0.8,$$

$$\text{Enhanced}(p_I) = \text{HSVtoRGB}(H(p_I), \text{EnhancedS}(S(p_I), V(p_I)).V(p_I)),$$

where H, S, and V are HSV channels of a pixel.

For **Black and White**:

$$\text{Enhanced}(p_I) = \text{RGB}(\text{Lum}(p_I), \text{Lum}(p_I), \text{Lum}(p_I)).$$

Tone and Color Curves. We use a differentiable piecewise-linear mapping function to represent curves, as detailed in the main paper. For tone curves, the same curve is applied to the image, and the slope of each segment in the curve is in $[0.5, 2.0]$. For color, a separate curve is applied to each of the three color channels, with slopes in $[0.9, 1.1]$. The bounds on the curve slopes reflect the fact that human artists do not usually apply sharp color curves, but sometimes may use a strong tone curve.

Experimental Details

MIT-Adobe FiveK Dataset Partitions. The MIT-Adobe FiveK dataset is randomly separated into three parts, which are listed in the data files **FiveK_train1.txt**, **FiveK_train2.txt** and **FiveK_test.txt**. For the test set, we select 100 random images employed in the user study on AMT, as listed in file **FiveK_test_AMT.txt**.

Histogram Intersection Details. The quantities for histogram intersection are defined as follows:

- Luminance is defined as the mean pixel luminance (defined previously as **Lum**.)
- Contrast is defined to be twice the variance of pixel luminance.
- Saturation is defined as the mean pixel saturation (the "S" value in the HSL color space).

The results are separated into 32 equal bins within the interval $[0, 1]$, i.e. $[0, 1/32), [1/32, 2/32), \ldots$

However, with only 1,000 sample images, only about 31.25 images will be placed in each bin on average, resulting in significant measurement noise. Therefore, we augment the data for histogram intersection by cropping 16 patches in each image, and measure the histogram quantities on these 16,000 image patches. Please refer to the accompanying code (**histogram_intersection.py**) for the detailed algorithm on measuring this error metric.

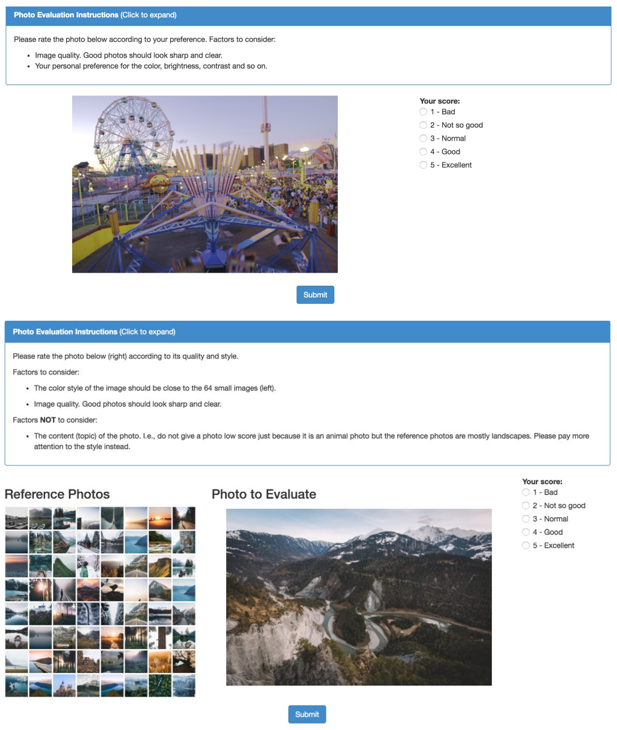


Photo Evaluation Instructions (Click to expand)

Please rate the photo below according to your preference. Factors to consider:

- Image quality. Good photos should look sharp and clear.
- Your personal preference for the color, brightness, contrast and so on.

Your score:
- 1 - Bad
- 2 - Not so good
- 3 - Normal
- 4 - Good
- 5 - Excellent

Photo Evaluation Instructions (Click to expand)

Please rate the photo below (right) according to its quality and style.

Factors to consider:

- The color style of the image should be close to the 64 small images (left).
- Image quality. Good photos should look sharp and clear.

Factors **NOT** to consider:

- The content (topic) of the photo. I.e., do not give a photo low score just because it is an animal photo but the reference photos are mostly landscapes. Please pay more attention to the style instead.

Your score:
- 1 - Bad
- 2 - Not so good
- 3 - Normal
- 4 - Good
- 5 - Excellent

Fig. 19. Our AMT UIs for user studies.

Amazon Mechanic Turk. The AMT interfaces for evaluation are shown in Figure 19.

Human performance measurement. We present the users a short video (with subtitles) demonstrating how our software should be used. The user studies take about 3 minutes per image (roughly 30 minutes for each user to retouch 10 images). We do not enforce any time limit on the task. All 10 users are highly educated and their ages range from 20 to 30.

Scalability in Resolution. The ability to process high-resolution images is critical in professional photography. In Figure 20, 21 and 22, we show high-resolution results from our method, Pix2pix, and CycleGAN. It is clear that our method produces images with the highest quality on high-resolution images.

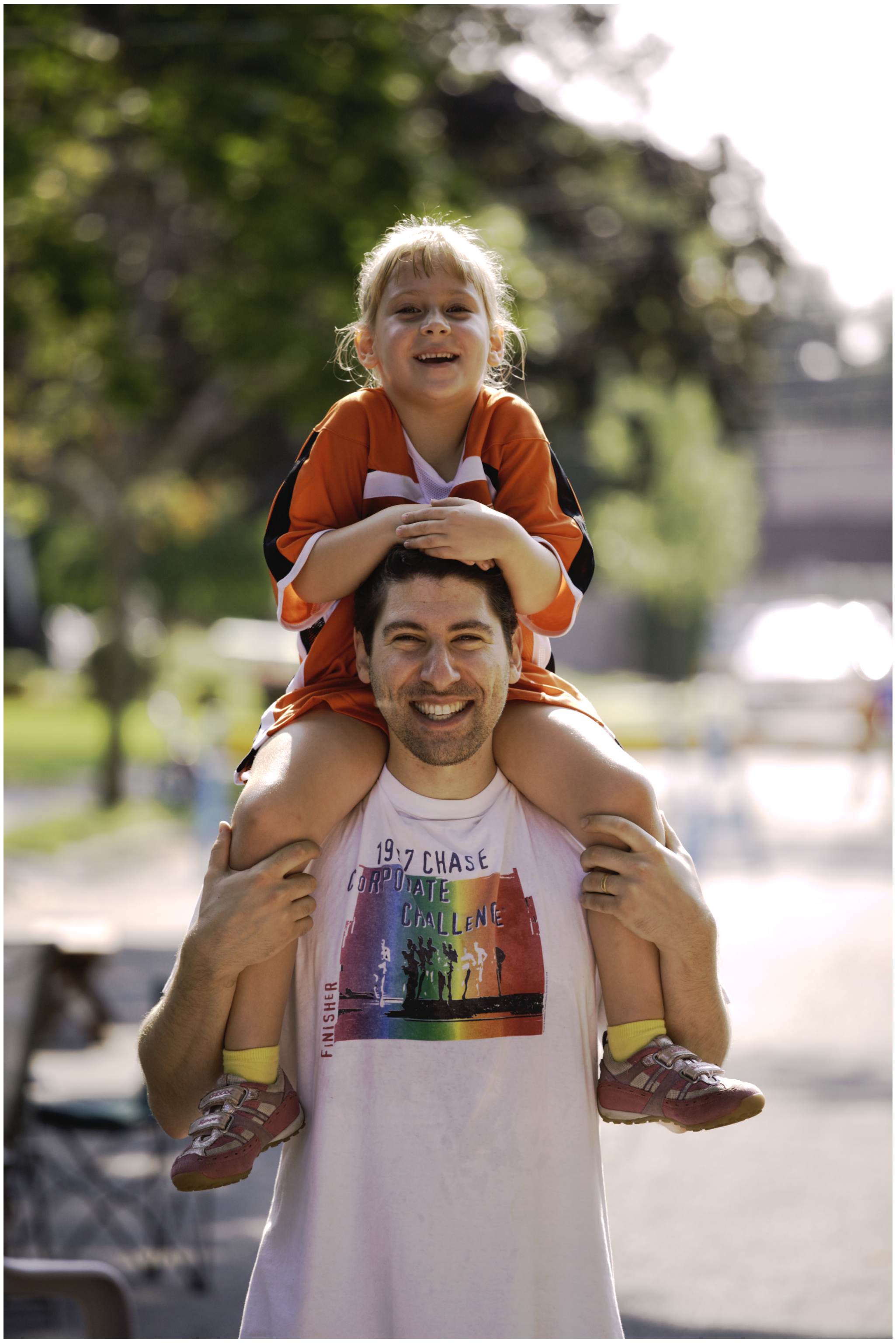


Fig. 20. Our method can cleanly handle images of any resolution.

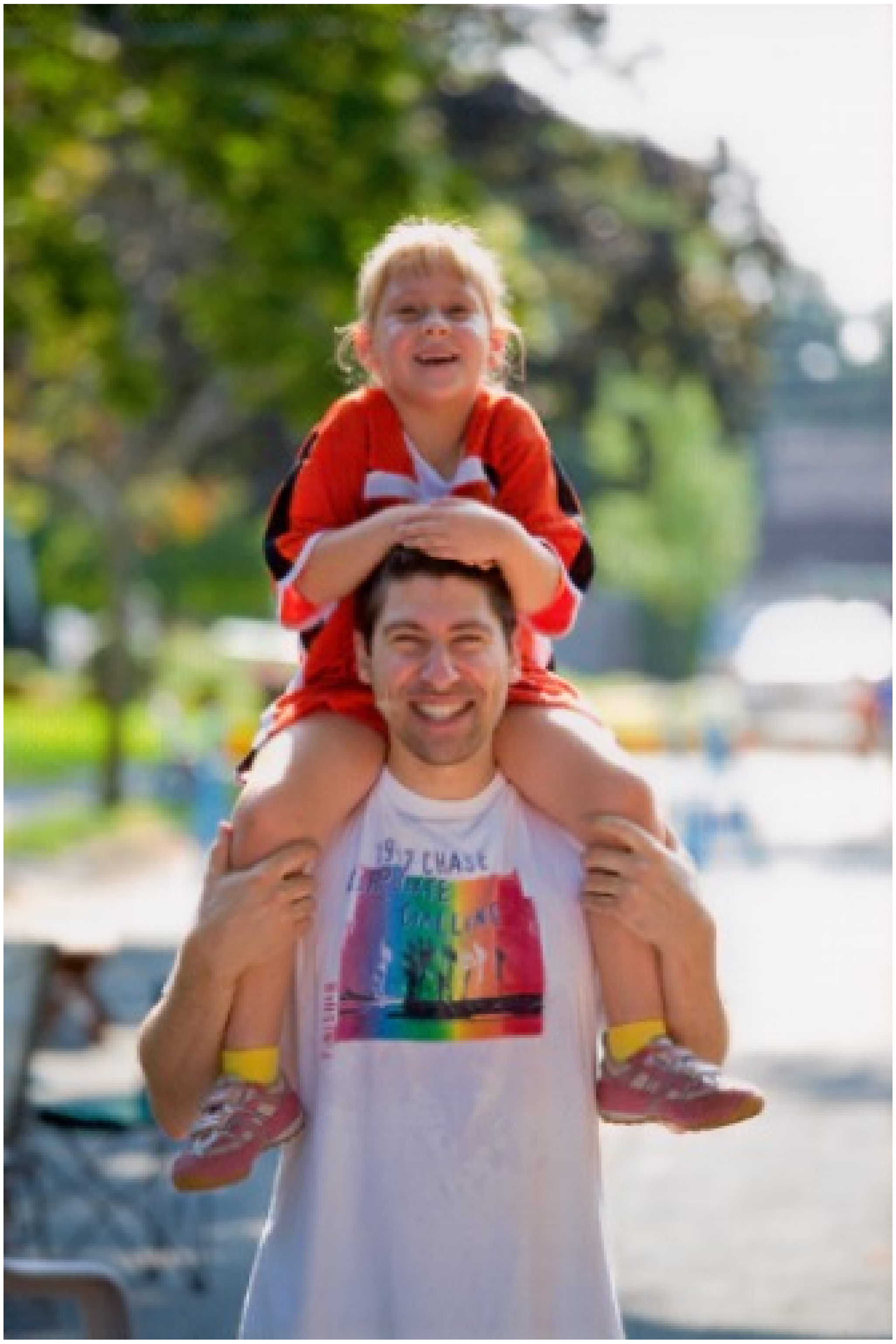


Fig. 21. Pix2pix result.

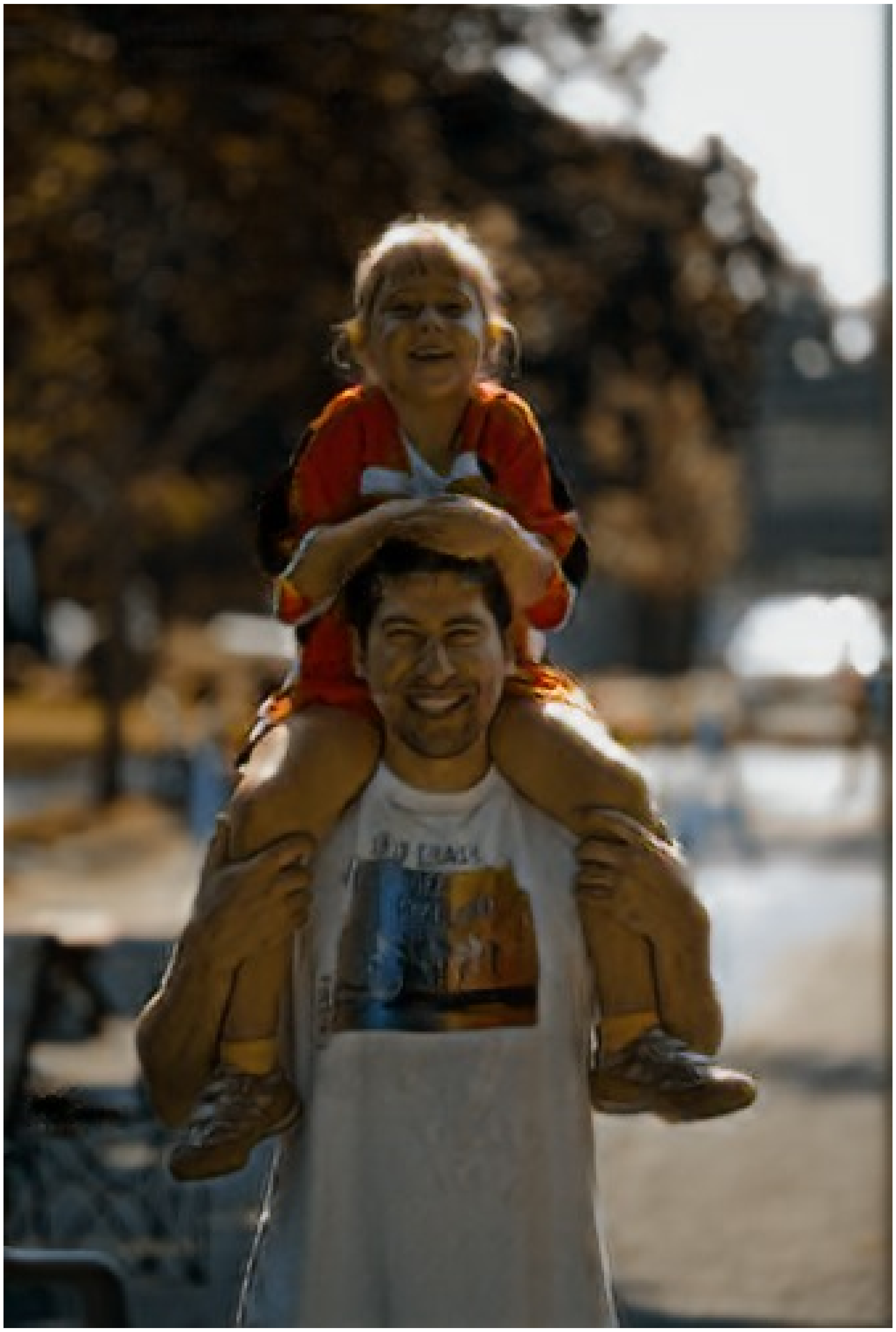


Fig. 22. CycleGAN result.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

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

Kevin Dale, Micah K. Johnson, Kalyan Sunkavalli, Wojciech Matusik, and Hanspeter Pfister. 2009. Image restoration using online photo collections. In *International Conference on Computer Vision*.

Alexei Efros and Thomas Leung. 2014. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*.

Chen Fang, Zhe Lin, Radomir Mech, and Xiaohui Shen. 2014. Automatic image cropping using visual composition boundary simplicity and content preservation models. In *ACM Multimedia*. 1005–1008.

Hui Fang and Meng Zhang. 2017. Creatism: A deep-learning photographer capable of creating professional work. *arXiv preprint arXiv:1707.03491* (2017).

William T. Freeman, Thouis R. Jones, and Egon C. Pasztor. 2002. Example-based super-resolution. 22, 2 (2002), 56–65.

Michael Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Fredo Durand. 2017. Deep bilateral learning for real-time image enhancement. 36, 6 (2017).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028* (2017).

Kan Guo, Dongqing Zou, and Xiaowu Chen. 2016. 3D Mesh Labeling via Deep Convolutional Neural Networks. 35, 1 (2016).

Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. 2016. Burst photography for high dynamic range and low-light imaging on mobile cameras. 35, 6 (2016).

Sungju Hwang, Ashish Kapoor, and Sing Bing Kang. 2012. Context-based automatic local image enhancement. In *European Conference on Computer Vision*.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004* (2016).

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192* (2017).

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Diederik Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2016. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Advances in neural information processing systems*.

Joon-Young Lee, Kalyan Sunkavalli, Zhe Lin, Xiaohui Shen, and In So Kweon. 2016. Automatic content-aware color and tone stylization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2470–2478.

Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. Dissertation. Fujitsu Laboratories Ltd.

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in neural information processing systems*. 469–477.

Ziwei Liu, Lu Yuan, Xiaoou Tang, Matt Uyttendaele, and Jian Sun. 2014. Fast burst images denoising. 33, 6 (2014).

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context Encoders: Feature Learning by Inpainting. In *Computer Vision and Pattern Recognition*.

Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. 2015. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 80.

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2016. Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning. 35, 4 (2016).

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

Xue Bin Peng and Michiel van de Panne. 2017. Learning locomotion skills using DeepRL: does the choice of action space matter?. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 12.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. 2016. Learning from Simulated and Unsupervised Images through Adversarial Training. *arXiv preprint arXiv:1612.07828* (2016).

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 387–395.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

Baoyuan Wang, Yizhou Yu, and Ying-Qing Xu. 2011. Example-based image color and tone style enhancement. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 64.

Xiaolong Wang and Abhinav Gupta. 2016. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*.

Jianzhou Yan, Stephen Lin, Sing Bing Kang, and Xiaoou Tang. 2014. A learning-to-rank approach for image color enhancement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2987–2994.

Jianzhou Yan, Stephen Lin, Sing Bing Kang, and Xiaoou Tang. 2015. Change-based image cropping with exclusion and compositional features. *International Journal of Computer Vision* 114, 1 (2015), 74–87.

Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. 2016. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)* 35, 2 (2016), 11.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European Conference on Computer Vision*. Springer, 649–666.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv preprint arXiv:1703.10593* (2017).