STRUCT Paper Reading

Exposure: A White-Box Photo Post-Processing Framework

ACM Transaction on Graphics（4.088）Accepted

Yuanming Hu, Massachuse s Institute of Technology
Hao He, Massachuse s Institute of Technology
Chenxi Xu, Peking University
Baoyuan Wang, Microsoft Research
Stephen Lin, Microsoft Research

Wenjing Wang
18.01.28

CATALOG

BACKGROUND

- Automatic Photo Retouching
- The state-of-the-art methods：
  - mainly based on supervised learning from paired images （by an expert photographer）
  - handcrafted features

BACKGROUND

- CNNs:
- Automatic Photo Adjustment Using Deep Neural Networks: DNN and handcrafted global and local features

Input Image Ground Truth Our Result

- Deep Bilateral Learning for Real-Time Image Enhancement: predict local affine transforms in bilateral space, edge-aware

12 megapixel 16-bit linear input (tone-mapped for visualization)

tone-mapped with HDR+ **400 – 600 ms**

processed with our algorithm **61 ms**, PSNR = **28.4 dB**

BACKGROUND

- Automatic Content-Aware Color and Tone Stylization [Lee et al. 2016]
  - Color and Tone Stylization, unsupervised
  - finding exemplar images whose color and tone style is compatible with a given image

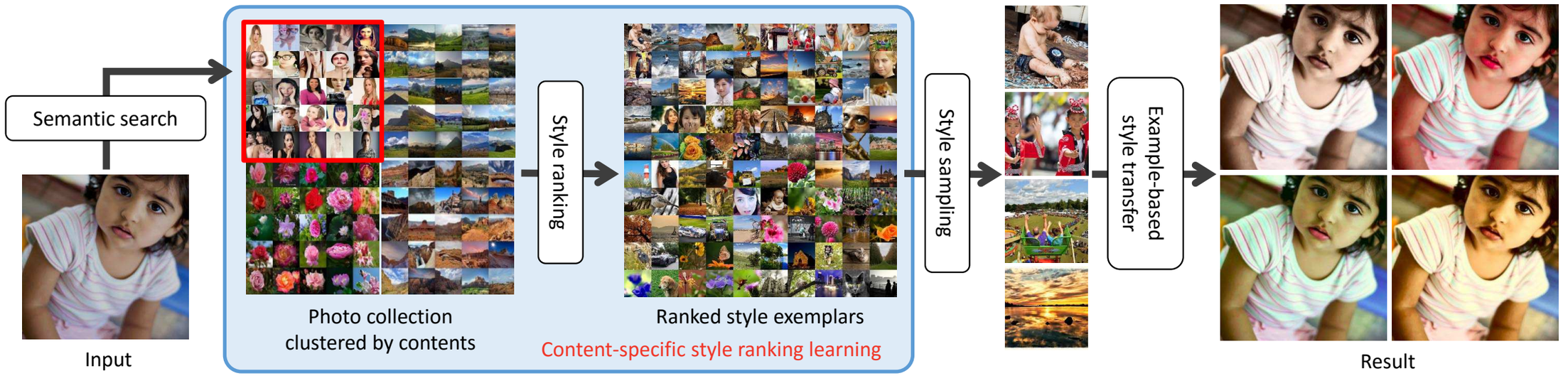


Figure 2. The overall framework of our system.

CATALOG

PURPOSE & EFFECT

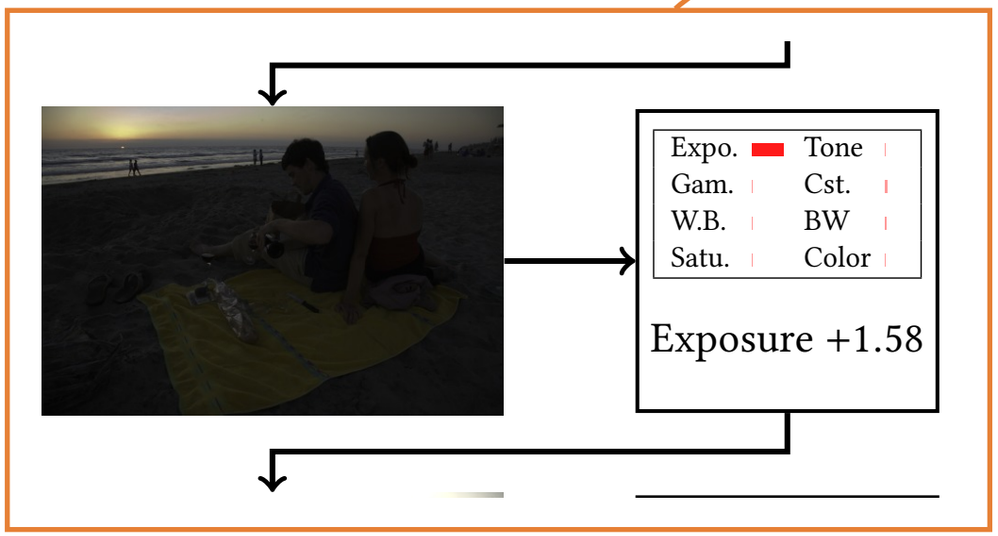
- Contributions:
  - **Understandable:** network's chooses can be understood and reproduced by users, **rather a black-box**.
  - **Without pairs:** without paired image data.
  - Transform into **style of a photo collection:** Instead of converting into a specific result, learns to transform an image into a certain style as represented by a photo collection.

RL
(making decision)

GAN

PURPOSE & EFFECT

- RL: Post-processing as a decision-making sequence



- Make decisions on:
  - Filter?
    Gamma\Exposure\Contrast…
  - parameter?

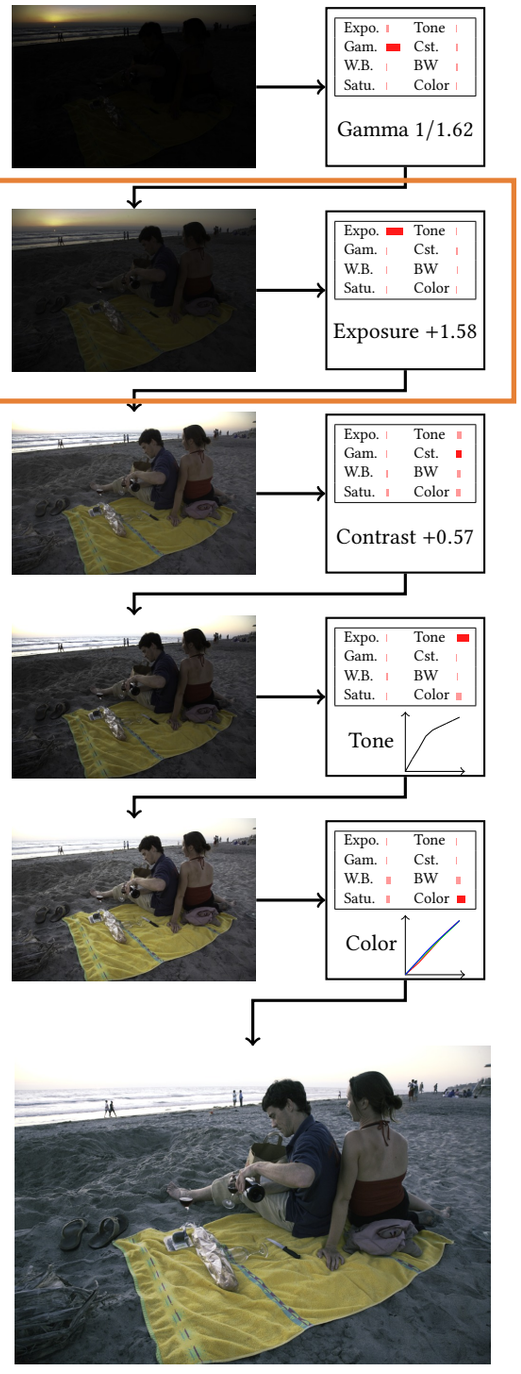


Fig. 11. Example of a learned retouching operation sequence from artist A (500px).

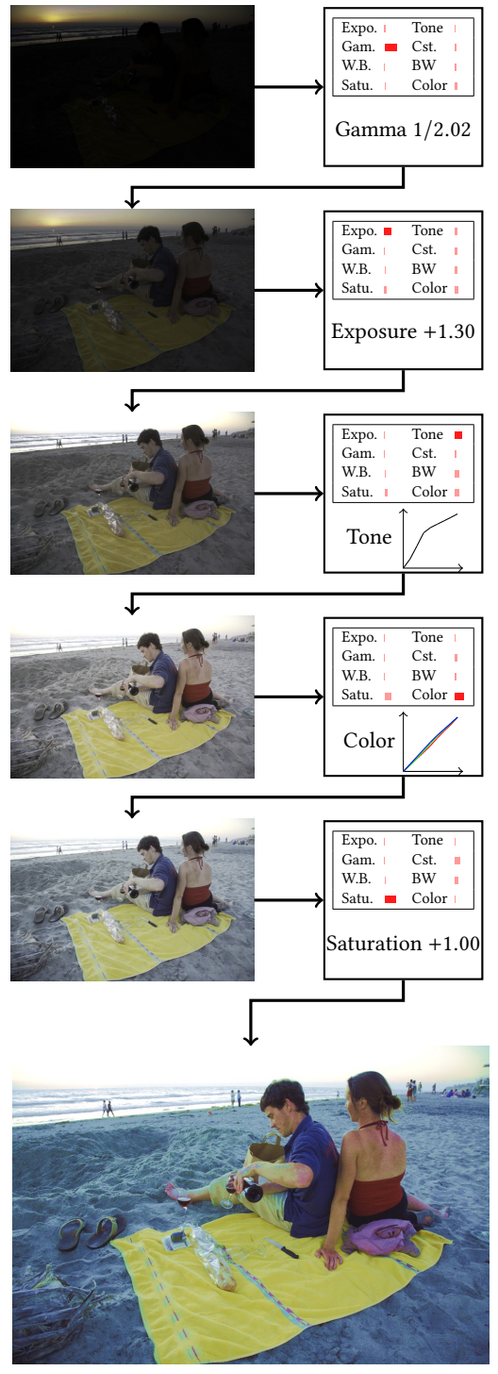


Fig. 12. Example of a learned retouching operation sequence from artist B (500px). Note that different from artist A, photos from artist B are more saturated, which is reflected in this learned operation sequence.

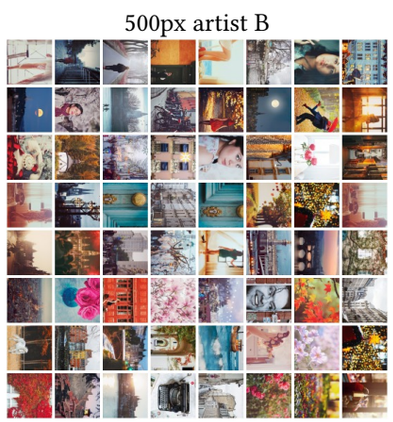
PURPOSE & EFFECT

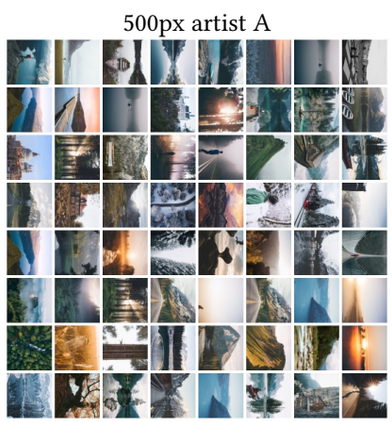
- GAN(WGAN):
- discriminator tells if the image is from the target dataset or was generated by the generator

paired:

为了和*pix2pix*等比较
但是训练的时候仍然是无*pair*

not paired:





PURPOSE & EFFECT

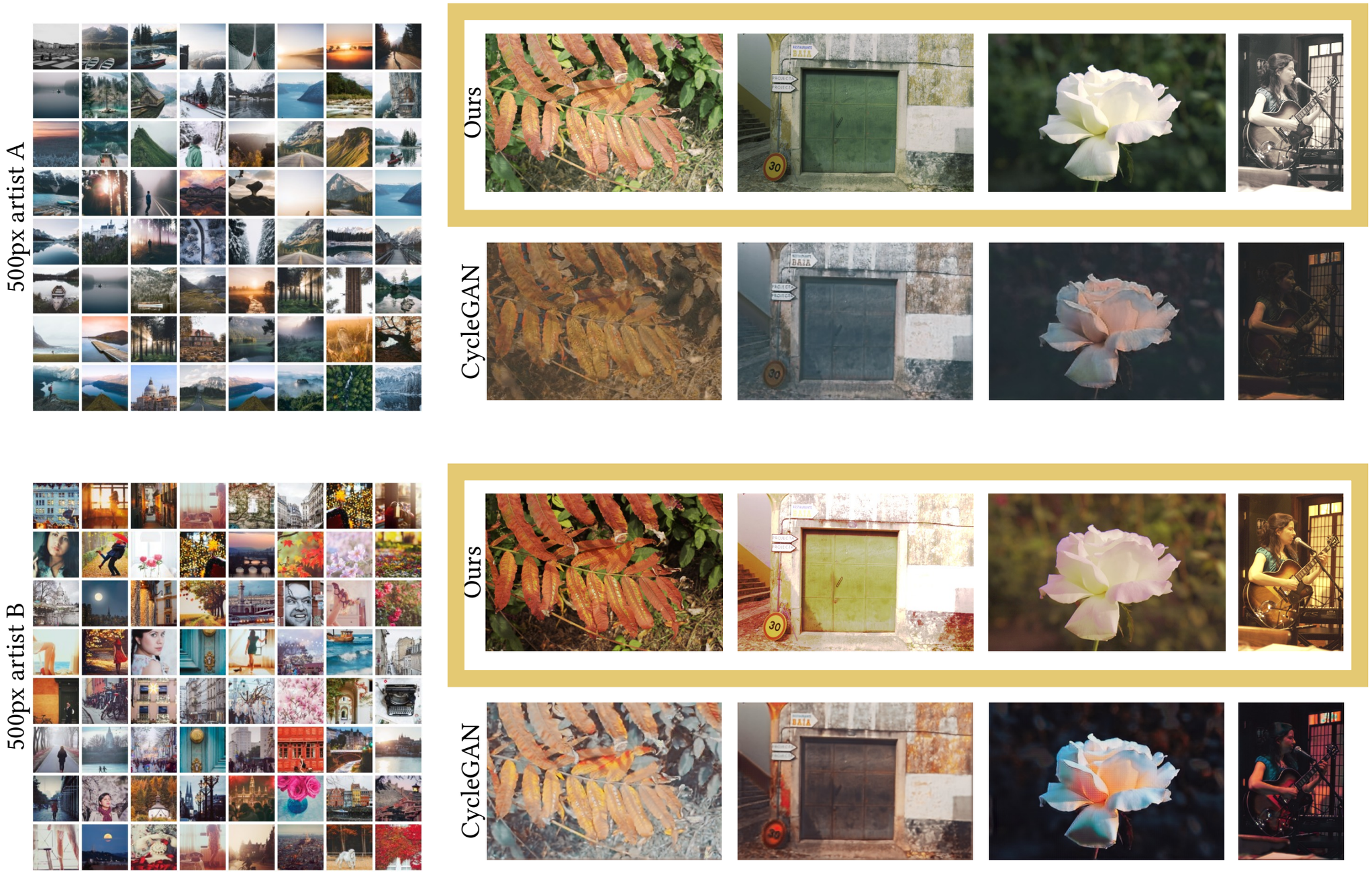


Fig. 10. Learning the styles of two artists from 500px.com, using our system and CycleGAN.

CATALOG

- Author
- Background
- Purpose & Effect
- Model
- Experiment

RL PART —— MODEL

- Problem: **P = (S,A)**. **S** is state space（RAW input and all intermediate results）, **A** is action space（filter operations, e.g. exposure, white balance, contrast）
- Reward function **r**
- return $\mathbf{r_k}$:

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

  where γ： discount factor, places greater importance on rewards in the nearer future.

- Policy **π**

RL PART —— MODEL

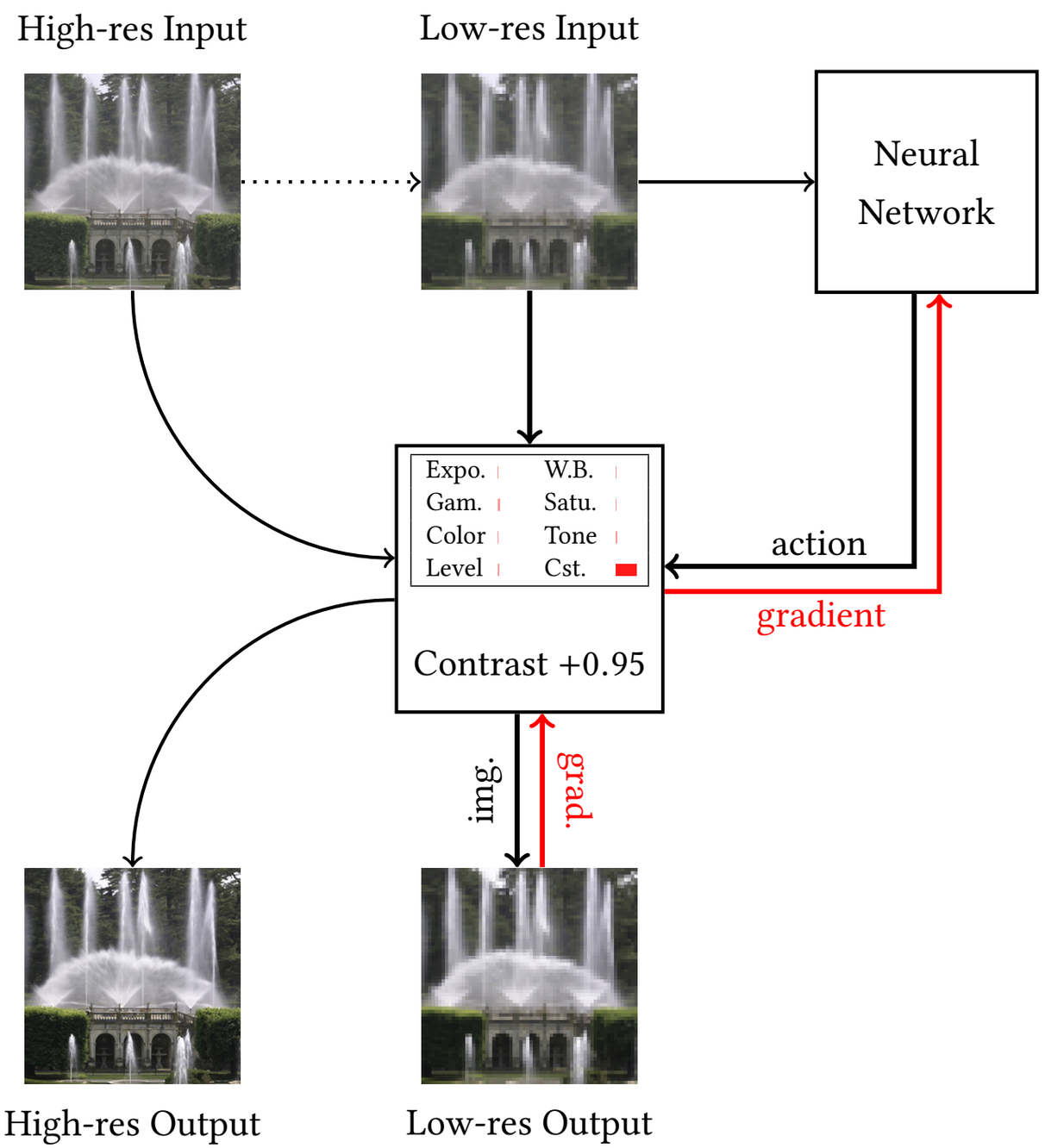
- Hope to maximize objective **J(π)** (s_0 is input) :

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

- decompose actions into two parts:
  - a_1 : select filters
  - a_2 : a continuous decision on filter parameters
- decompose policy into two parts:
  - π_1 : takes (s) and decide a_1
  - π_2 : takes (s,a_1) and decide a_2

RL PART ——PRINCIPLES

- Filter should be:
- **Differentiable**: for back propogation.
- **Resolution-independent**: so that we can train on 64×64 px and run the model on 6000×4000 px.
- **Understandable**: which means filters should have intuitive meaning。



RL PART —— FILTER DESIGN

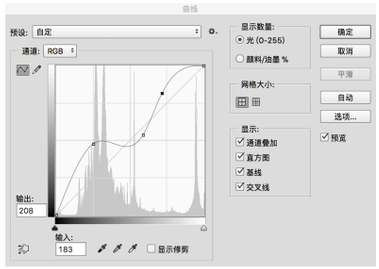
➤ input $P_I = (r_I, g_I, b_I)$ → output $P_O = (r_O, g_O, b_O)$

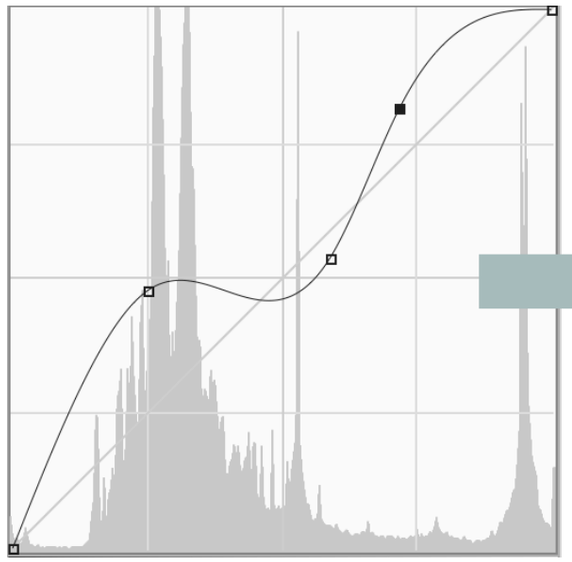
Operation	Parameters	Filter
Exposure	E	$P_O = 2^E P_I$
White Balance	W_r, W_g, W_b	$P_O = (W_r r_I , W_g g_I , W_b b_I)$
Color curves	$C_{i,k}$	$P_O = (L_{Cr}(r_I), L_{Cg}(g_I) , L_{Cb}(g_I))$

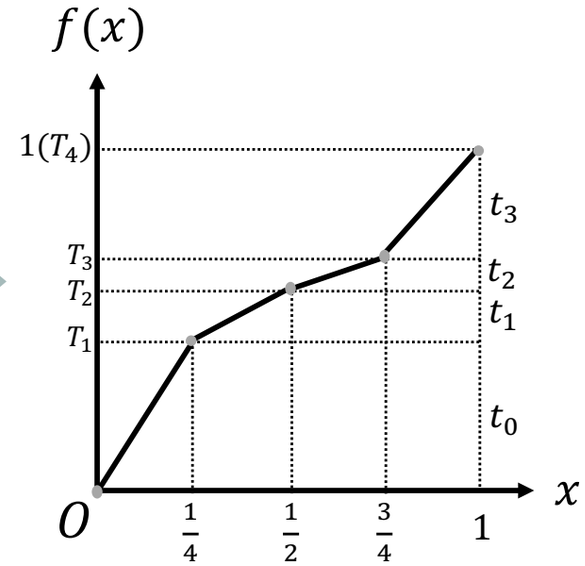
requires special treatment for its filter to be differentiable

RL PART —— FILTER DESIGN

- Curve（曲线）：







- using L parematers: $\{t_0,t_1,\ldots,t_{L-1}\}$, $T_k = \sum_{i}^{k-1} t_i$, points on the curves are represented as $(k/L, T_k/T_L)$

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

- find that $L = 8$ is sufficeint.

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

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 $\text{Lum(p)} = 0.27p_r + 0.67p_g + 0.06p_b$.

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**:

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

RL PART —— FILTER DESIGN

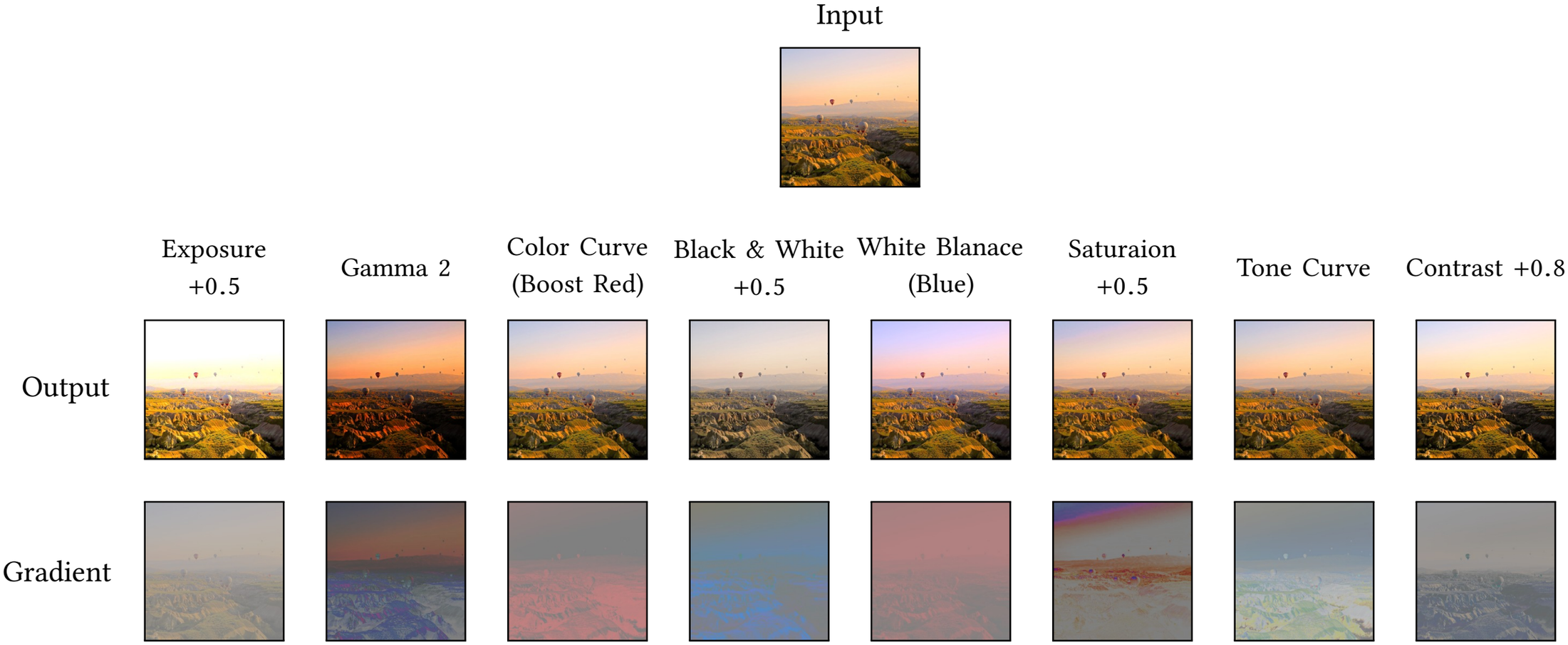
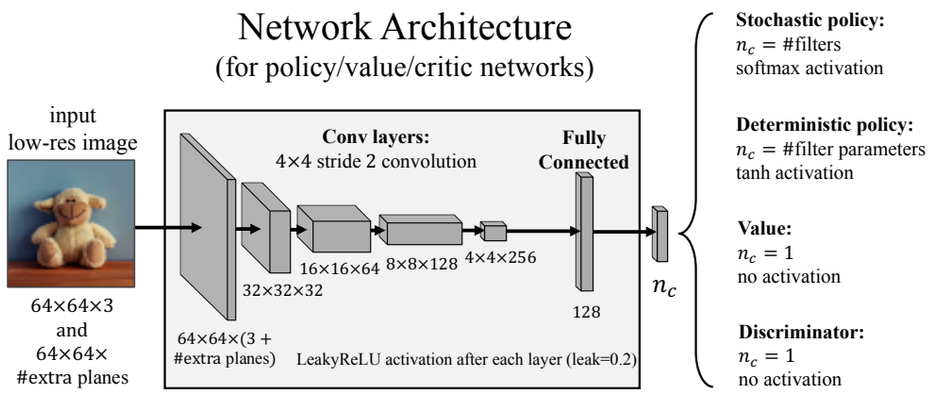


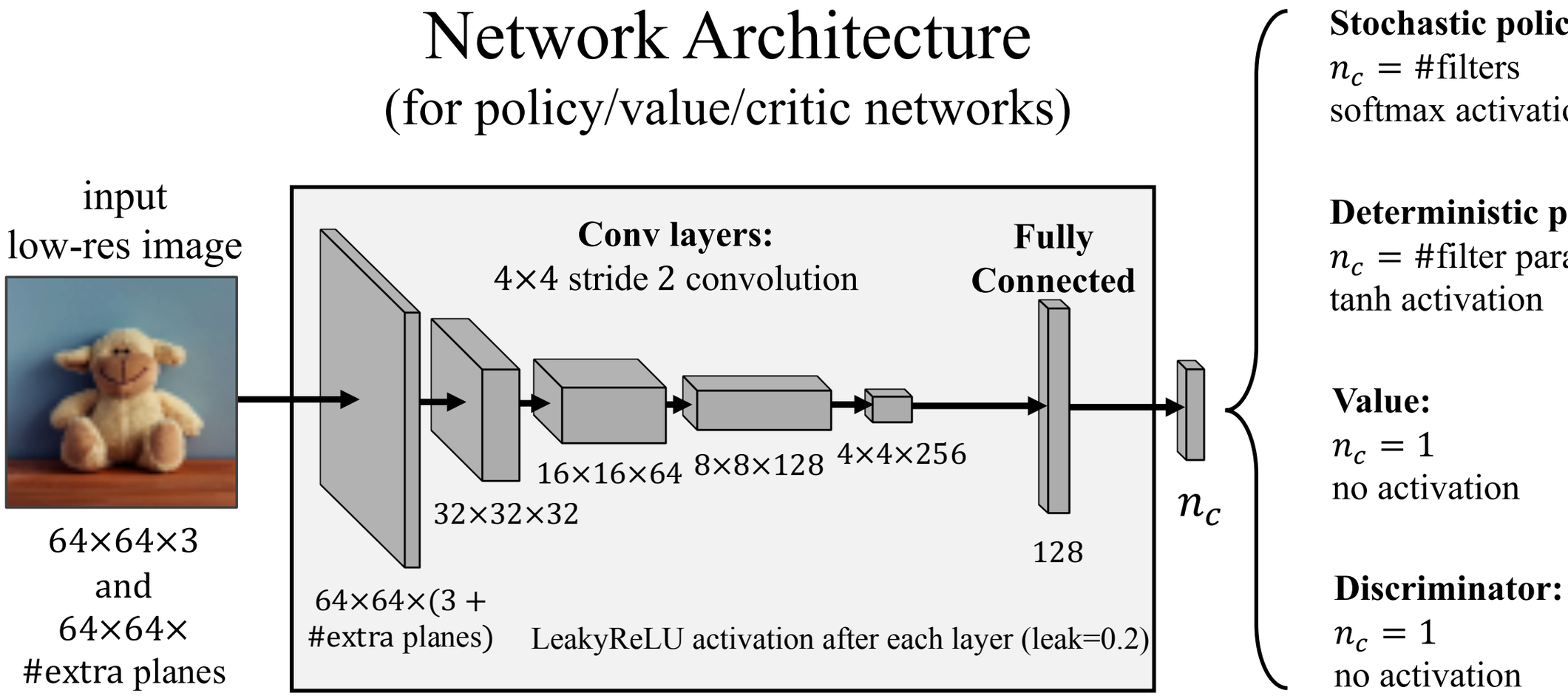
Fig. 5. Visualizations of the eight differentiable filters. Gradients are displayed with a +0.5 offset so that negative values can be properly viewed. For the white balance filter, we visualize the gradient with respect to the red channel parameter; for the tone/color curves, we differentiate with respect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

RL PART —— LEARNING

- Two policy networks
  - map the images into action probabilities π_1 (after softmax)
  - map the images into filter parameters π_2 (after tanh)
- value（RL，actor-critic algorithm ）
- discriminator（GAN）
- share basically the same architectre:



RL PART —— LEARNING



- Convolution: 4×4 and stride 2
- the first FC: reduce the number of outputs to 128
- the final FC: get parameters we need
- Naively using CNNs results in unsatisfactory learning of agent policies and global statistics. Therefore, we concatenate extra (spatially constant) feature planes as additional color channels in the input.

RL PART —— LEARNING

- For π1:
  - **Softmax** → which filter to use
  - extra feature planes(64×64×3): average luminance, contrast and saturation
- For π2 (one for each filter share the convolutional layers):
  - **Tanh** → filter parameters
  - extra feature planes(64×64×9): 8 boolean (which filters have been used) & the number of steps that have been taken
- For Value Net (actor-critic RL architecture):
  - extra feature planes(64×64×9): 8 boolean (which filters have been used) & the number of steps that have been taken

RL PART —— LEARNING

- Policy network training:
  - $J(\pi_1)$: Monte-Carlo policy gradient
  - $J(\pi_2)$: Deterministic policy gradient
  - using actor-critic algorithm to calculate
- Set the discount factor as $= 1$
- allow the agent to make 5 edits to the input image

GAN PART —— LEARNING

- WGAN + gradient penalty

TRAINING STABILIZATION

- we utilized the following strategies:
- **Exploitation vs.exploration:** choices between (i) devoting more attention on improving the current policy (ii) trying a new action in search of potentially greater future reward
- Penalize π_1 if its proposal distribution is too concentrated

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

- if the agent uses a filter twice, the second usage will incur an additional penalty of -1.

TRAINING STABILIZATION

- we utilized the following strategies:
- **Experience Replay:** This strategy helps to reduce sample correlation and stabilizes training

CATALOG

IMPLEMENTATION DETAILS

- On RAW images
- TensorFlow, code not available
- Training: less than 3 hours for all experiements
- Testing: unoptimized version takes 30ms for inference on an NVIDIA TITAN X (Maxwell) GPU
- Model size: < 30MB

DATASET

- The MIT-Adobe FiveK Dataset(Paired)
  - 5000 RAW images, retouched versions of five experts.


(a) input


(b) Retoucher A


(c) Retoucher B


(d) Retoucher C

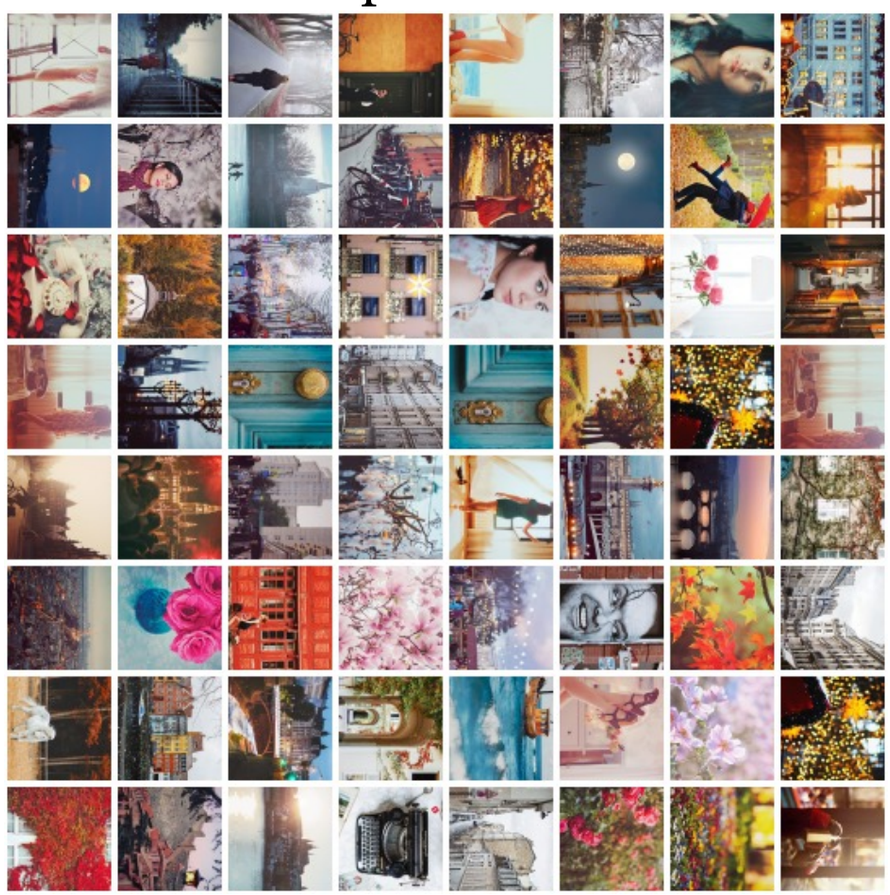

(e) Retoucher D


(f) Retoucher E

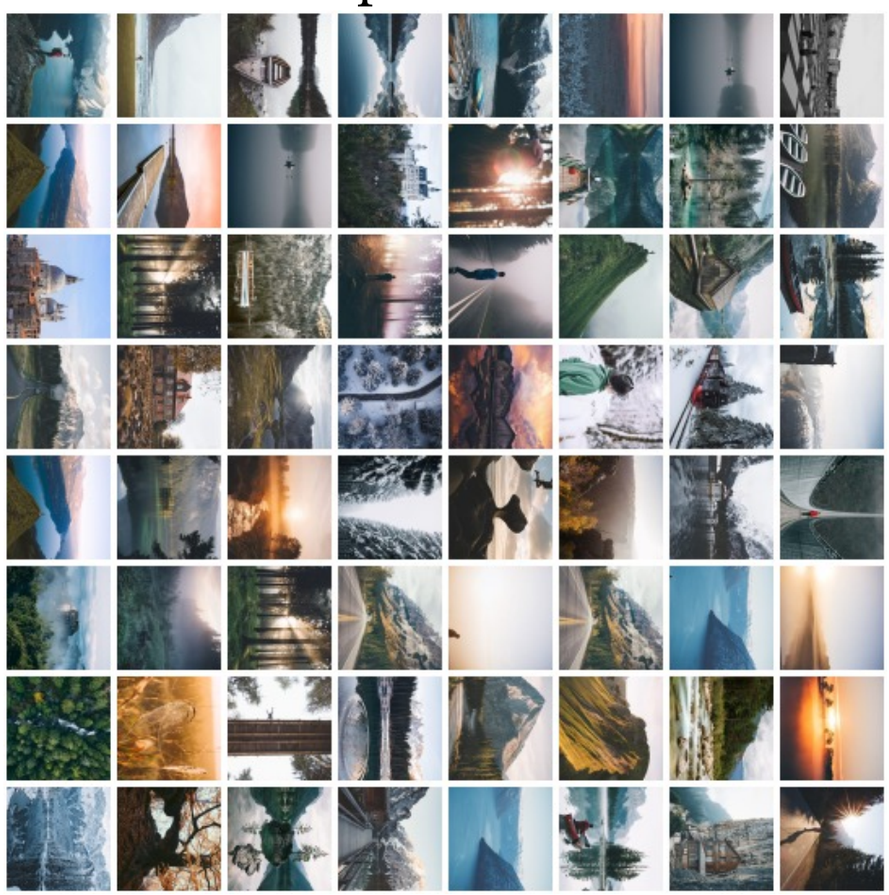
DATASET

- 500px.com (Unpaired)
  - professionally retouched photos from two artists, 369 from one and 397 from the other.

500px artist B

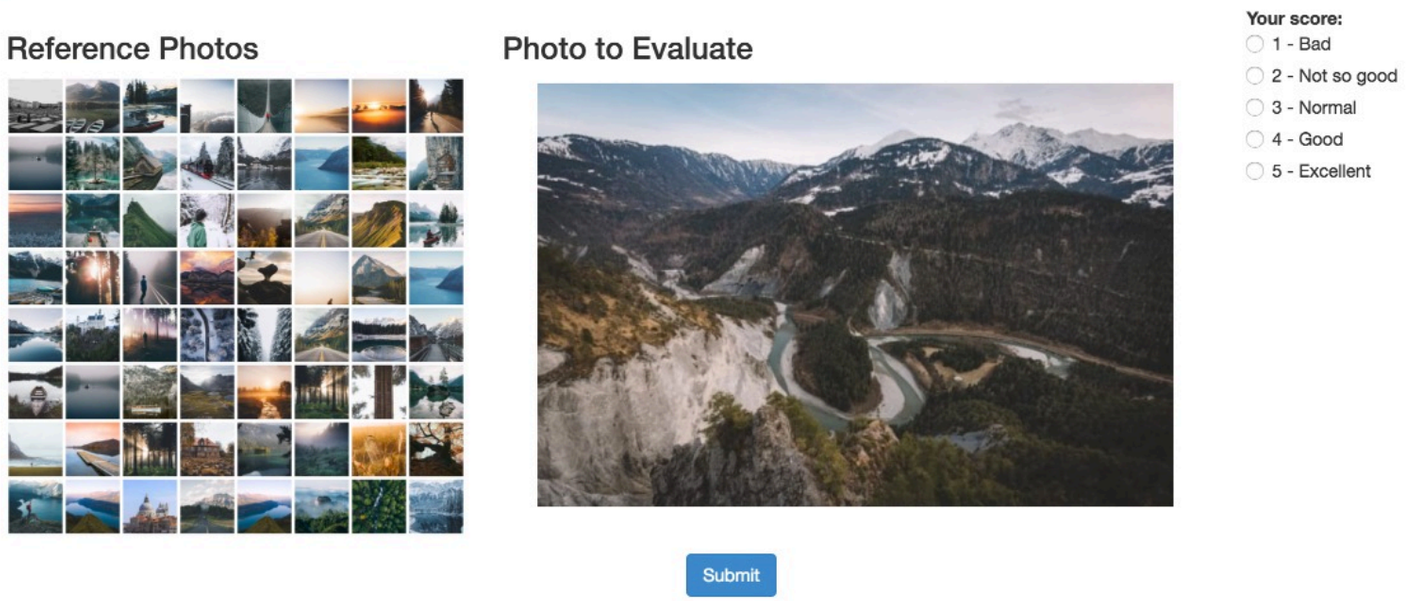


500px artist A



ERROR METRICS

- distributions of image properties
  - histograms of luminance, contrast, saturation (measure the distance in the output and target)
- user study
  - Amazon Mechanical Turk. Randomly choose 100 images and ask users to rate the image.



COMPARE

- CycleGAN: training on unpaired data, having limitation on resolution. Training takes 30 hours for 500 × 333 px.
- Pix2pix: training on paired data, having limitation on resolution.

Human (expert) CycleGAN Pix2pix Human (novice) Our method

Input

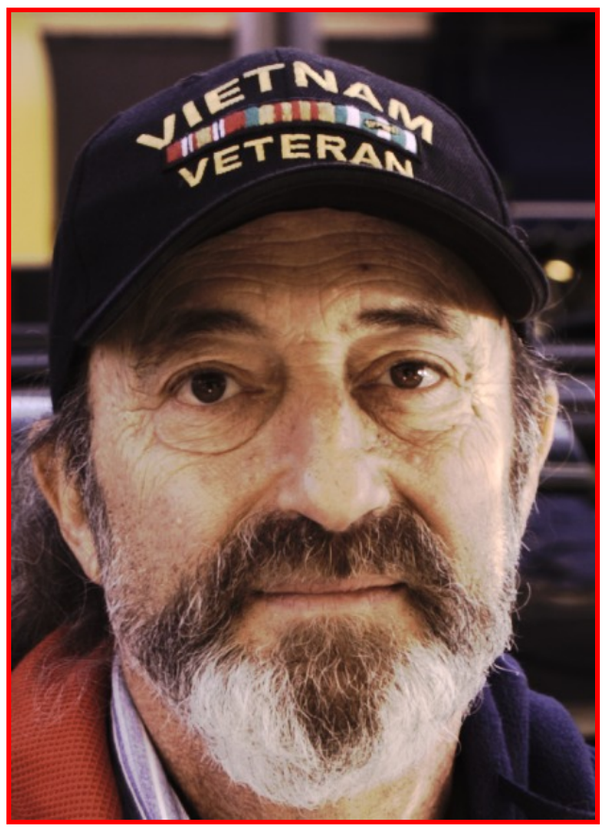
CycleGAN

Pix2pix

Ours

CycleGAN

Pix2pix



Ours

COMPARE

REASON:

receptive field (1/3 of the whole image width) is too small to 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

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

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

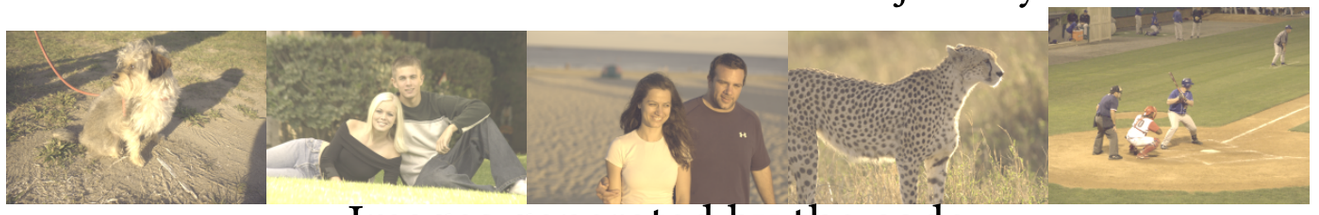
Approach	Histogram Intersection			User Rating
	Luminance	Contrast	Saturation	
Ours	85.2%	91.7%	83.5%	3.22 ± 0.04
CycleGAN	60.1%	79.4%	83.4%	2.86 ± 0.04
500px artist B	100%	100%	100%	3.40 ± 0.04

APPLICATION

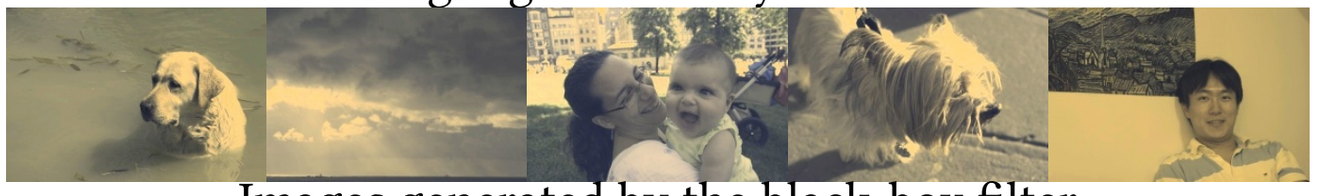
- Change fixed filters into code:

```python
# 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.

FAILURE CASES

- Cannot denoise: pixel-level denoising, may be challenging to model as resolution-independent, differentiable filters
- faces, images with poor content, composition or lighting conditions

SUMMARY

- End-to-end model, differentiable filters
- Undertandable
- Without image pairs
- Transform into style of images collections

- GAN scales with image resolution and generates no distortion artifacts in the image.
- May replace the actor-critic RL architecture and WGAN with other related alternatives.