Learning by Planning: Language-Guided Global Image Editing
Appendix

Jing Shi[1] Ning Xu[2] Yihang Xu[1] Trung Bui[2] Franck Dernoncourt[2] Chenliang Xu[1]
[1]University of Rochester [2]Adobe Research
[1]{j.shi,chenliang.xu}@rochester.edu [1]yxu74@u.rochester.edu [2]{nxu,bui,dernonco}@adobe.com

Contents

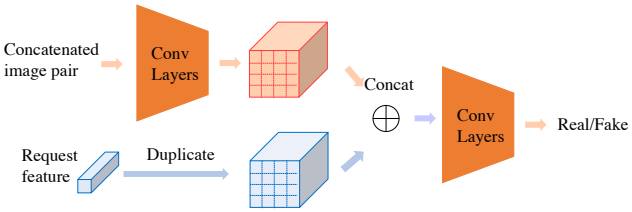

Figure 1. Structure of the discriminator used for adversarial loss.

A. Ablation Study

A.1. Different Image Loss

Inspired by the GAN-based image-to-image translation [12], we also try to apply adversarial loss $\mathcal{L}_{adv}$ using discriminator D, whose structure is shown in Fig. 1. The adversarial loss is expressed as:

$$\begin{aligned}\mathcal{L}_{adv} = &- \mathbb{E}_{(I_0,I_g)}[\log(D(I_0, I_g, Q))] \\ &- \mathbb{E}_{(I_0,I_T)}[\log(1 - D((I_0, I_T, Q)))]. \end{aligned} \quad (1)$$

Denote the whole parameter for the T2ONet as Θ_G and discriminator as Θ_D, the objective for adversarial loss is $\min_{\Theta_G}(\max_{\Theta_D} \mathcal{L}_{adv})$. The effect of L1 and adversarial loss is shown in Tab. 3. We observe that adding the image level loss can significantly improve T2ONet, because the operation supervision is trained in teacher forcing fashion, which easily accumulates error at each step. A supervision at the final image help correct the error at the final image. And without image supervision, the variance drops significantly, indicating the model has a very similar output for different requests. Moreover, the L1 loss is better than the adversarial loss. It might because adversarial loss is good at generating sharper and more detailed images [2, 5], but our operation will not reduce the detail/texture of the image, so

L1	Adv	L1↓	SSIM ↑	FID ↓	$\sigma_{\times 10^2}$↑
✗	✗	0.0949	0.8300	8.2482	0.0532
✓	✗	**0.0784**	0.8459	**6.7571**	**0.7190**
✗	✓	0.0901	0.8031	9.4600	0.5825
✓	✓	0.0801	**0.8464**	6.9436	0.5671

Table 1. Ablation study of different losses and network structures on the MA5k-Req test set. L1, Adv represent L1 and adversarial loss, respectively.

h	L1↓	SSIM ↑	FID ↓	$\sigma_{\times 10^2}$↑
0	**0.0784**	**0.8459**	**6.7571**	0.7190
0.01	0.0809	0.8487	7.2789	1.1008
0.1	0.0979	0.8090	8.8763	**2.1482**

Table 2. L1 and variance trade-off by training with different parameter sampling variance (reflected by h) on the MA5k-Req test set.

	L1↓	SSIM ↑	FID ↓	$\sigma_{\times 10^2}$↑
w./o. image	0.0863	0.8332	7.7869	**1.1950**
w./o. operation	0.0837	0.8424	7.6559	0.3257
w./o. attention	0.1088	0.8087	8.4587	0.8872
full model	**0.0784**	**0.8459**	**6.7571**	0.7190

Table 3. W./o. image, operation, and attention indicate the T2OCell without using in the intermediate image, operation, and attention on MA5k-Req test set.

the adversarial loss may not help as much as L1 loss, which pushes the similarity of the generated image to target image in a more direct way. And the combination of L1 and adversarial loss is still weaker than solely L1 loss in general, probably because we directly use $\mathcal{L} = \mathcal{L}_{L1} + \mathcal{L}_{adv}$ and didn't fine-tune the balance weight. Hence, to facilitate our model design, we purely use L1 loss as the image loss. The visual comparison of different final image losses is shown in Fig. 2 and we find that without L1 loss or changing L1 to adversarial loss, the visual appearance is less similar to target and less appearing.

A.2. Trade-off between L1 and Variance

Tab. 2 shows the complete evaluation for the trade-off between L1 and variance.

A.3. Effect of historical operations, images and attention for T2ONet

Our standard T2O Cell takes in the previous operation and image. The comparison with only either of them is shown in Tab. 3, indicating that just image or operation performs no better than their combination. One exception is the variance for the only operation is better than combined, which means without the historical image as the feedback, the editing will be less controlled and be more diversified. Also, the attention mechanism help improve the performance according to Tab. 3.

A.4. Comparison of other possible planning method

Since our operation planning is based on greedy best-search, such greedy method does not guarantee an optimal solution. Inspired by the ϵ-greedy policy [10] applied in RL, we further compare a variant called ϵ-greedy operation planning to incorporate randomness to further approach optimal. The only difference is that there is 5% possibility that the operation is randomly selected, rather than the top choice.

A.5. Effect of different single operation lists and different maximum operation steps

We further study the comparison of only applying single operation. Table 4 presents the editing results of planning and T2ONet using only single operation. The most effective operations are "tone" and "color", because they have 8 and 24 parameters, respectively, and thus have stronger editing ability than single-parameter operations . The results also suggest that the single-step editing results are worse than our proposed multi-step editing results. Also, we track the effect of different maximum operation steps, the planning results are shown in Tab. 5. From the perspective of planning, maximum steps 4, 5, 6 do not have much difference. This suggests that we could reduce the editing step or find the best trade-off between the editing effect and editing time complexity in the future.

B. Reinforcement Learning

B.1. Details of RL baseline

Now we reformulate the editing problem into a partial observed Markov decision process and introduce an RL baseline. Following the symbol notations and the problem formulation in the main paper that the editing process is a sequential action decision problem, we augmented with reward r_{t+1} for action a_t, the problem can be reformulated into a Markov decision process and solved by RL. Following [4], the reward is set to indicate the incremental image quality, which is adapted as the reduction of the image cost

$$r_t = \text{cost}(I_{t-1}) - \text{cost}(I_t), \tag{2}$$

where $\text{cost}(I)$ can be any image loss and is set as $||I - I_g||_1$ in our experiment. Since the reward for the "END" action is hard to design (the reward in Eq. (2) is zero for "END" action), we set every episode fixed T steps ($T = 5$ as [4]). The actions are sampled from policy $\pi = (\pi_o, \pi_\alpha)$, where $\pi_o = P(o|s), \pi_\alpha = P(\alpha|o, s)$, leading to the trajectory $\Pi = \{s_0, a_0, s_1, r_1, ..., s_T, r_T\}$. The $P(o|s), P(\alpha|o, s)$

Operation	Brightness	Contrast	Saturation	Sharpness	Tone	Color	Input
planning (train)	0.0521	0.0859	0.1037	0.1163	**0.0277**	**0.0260**	0.1202
T2ONet (test)	0.1315	0.1178	0.1163	0.1256	**0.1006**	**0.1129**	0.1190

Table 4. L1 distance to target image over different **single operations** on MA5k-Req dataset. Input represents the distance of the input image to the target image. Planning results are on the training set, and T2ONet results are on the testing set.

Max Step	1	2	3	4	5	6	Input
Planning (train)	0.0256	0.0145	0.0139	0.0137	0.0136	0.0136	0.1202

Table 5. L1 distance to the target image with different **maximum editing steps** on MA5k-Req dataset. Input represents the distance of the input image to the target image.

is computed the same way as T2ONet. With the accumulated reward defined as $G_t = \sum_{\tau=0}^{T-t} \gamma^\tau r_{t+\tau}$ ($\gamma = 1$ as [4]), the goal is to optimize the objective $J(\pi) = \mathbb{E}_{(I_0,Q)\sim P(\mathcal{D}),\Pi\sim\pi} G_1$, where $p(\mathcal{D})$ is the distribution of the dataset. Denoting θ_o and θ_α as the respective model parameter involving in the computation of o and α, the discrete policy π_o is optimized via REINFORCE [13]:

$$\nabla_{\theta_o} J(\pi) = \mathop{\mathbb{E}}_{\substack{(I_0,Q)\sim P(\mathcal{D}) \\ o_t\sim\pi_o, \alpha_t\sim\pi_\alpha}} \sum_{t=0}^{T-1} G_{t+1} \nabla_{\theta_o} \log \pi_o(o_t). \quad (3)$$

For the continuous policy π_α, we resort to DPG [9]. Different from the common setting [9, 4] where the Q function is approximated with a neural network to make it differentiable to action, we approximate Q as G since our G_{t+1} is already differentiable to α_t, resulting in the DPG as

$$\nabla_{\theta_\alpha} J(\pi) = \mathop{\mathbb{E}}_{\substack{(I_0,Q)\sim P(\mathcal{D}) \\ \alpha_t\sim\pi_\alpha}} \sum_{t=0}^{T-1} \nabla_{\alpha_t} G_{t+1} \nabla_{\theta_\alpha} \alpha_t. \quad (4)$$

In short, the major difference of our RL optimization from [4] is that we replace Q function approximated by neural network in [4] with G in both discrete and continuous policies, avoiding the complexity for training the Q network. The full algorithm for our RL baseline is in Appx. B.3.

In our experiments, the sampling for o is based on π_o with ϵ-greedy policy where the $\epsilon = 0.05$. The sampling for α is based on π_α where the gaussian width controller $h = 0.1$. The other implementation details are the same with our main experiments.

B.2. Equivalence of image loss and DPG

Now, we show the equivalence between image loss and DPG using the following theorem:

Theorem 1. *The DPG for α in Eq. (4) can be rewrite as*

$$\nabla_{\theta_\alpha} J(\pi) = -\mathop{\mathbb{E}}_{\substack{(I_0,Q)\sim P(\mathcal{D}) \\ \alpha\sim\pi_\alpha}} \frac{\partial \text{cost}(I_T)}{\partial \theta_\alpha}. \quad (5)$$

Proof. Substituting Eq. (2) and $\gamma = 1$, G_t can be simplified as

$$\begin{aligned} G_t &= \sum_{\tau=0}^{T-t} \big(\text{cost}(I_{t+\tau-1}) - \text{cost}(I_{t+\tau})\big) \\ &= \text{cost}(I_{t-1}) - \text{cost}(I_T). \end{aligned} \quad (6)$$

Since I_t is independent of α_t, we have

$$\nabla_{\alpha_t} G_{t+1} = \frac{\partial(\text{cost}(I_t) - \text{cost}(I_T))}{\partial \alpha_t} = -\frac{\partial \text{cost}(I_T)}{\partial \alpha_t}. \quad (7)$$

Therefore, the summation in Eq. (4) can be expressed as

$$\sum_{t=0}^{T-1} \nabla_{\alpha_t} G_{t+1} \nabla_{\theta_\alpha} \alpha_t = -\sum_{t=0}^{T} \frac{\partial \text{cost}(I_T)}{\partial \alpha_t} \frac{\partial \alpha_t}{\partial \theta_\alpha} = -\frac{\partial \text{cost}(I_T)}{\partial \theta_\alpha} \quad (8)$$

According to Eq. (8), Eq. (4) is equivalent to Eq. (5). □

B.3. Algorithm for RL baseline

The full algorithm for our RL baseline is shown in Alg. 1.

Algorithm 1: RL

Input: Training dataset $\mathcal{D}$; learning rate β; max operation step $N = 5$

for *episode in* $1 : M$ **do**
  Sample I_0, Q, I_g from $\mathcal{D}$;
  Sample one editing episode from π_o, π_α:
  $\{I_0, a_0, I_1, r_1, a_2, I_2, \ldots, I_T, r_T\}$;
  $\Delta_{\theta_o} J = \sum_{t=0}^{T-1} G_{t+1} \nabla_{\theta_o} \log \pi_o(o_t)$;
  $\theta_o \leftarrow \theta_o + \beta \Delta_{\theta_o} J$;
  $\Delta_{\theta_\alpha} J = -\frac{\partial \text{cost}(I_T)}{\partial \theta_\alpha}$; # $\text{cost}(I) = ||I - I_g||_1$
  $\theta_\alpha \leftarrow \theta_\alpha + \beta \Delta_{\theta_\alpha} J$;
end
return $(\theta_o, \theta_\alpha)$

Dataset	Pretrain	L1↓	SSIM ↑	FID ↓	$\sigma_{\times 10^2}$↑
MA5k-Req	✗	0.1007	0.8283	7.4896	1.6175
MA5k-Req	✓	0.0955	0.8330	7.1413	1.4672
GIER	✗	0.2286	0.3832	132.1785	0.3978
GIER	✓	0.1052	0.8075	49.4183	1.0949

Table 6. The RL performance with and without operation-supervised pretrain on two datasets.

B.4. Can operation planning benefit RL?

Since the success of RL relies on the exploration of the action space, can the action sequence obtained from the operation planning algorithm help RL to better explore the action space, especially the continuous action? To answer this question, similar to [15], we firstly pretrain the model with the planned operations as supervision (same as T2ONet training loss), then finetune it using RL with only the target image supervision. The result in Tab. 6 show that the pretraining does not help RL much on MA5k-Req, but significantly benefit RL on GIER. As GIER has smaller size and more complex editing than FiveK, RL is struggling with the exploration of α. The pretrained model can initialize a good exploration and thus the RL can work on GIER.

C. Planning for Local Editing

Our operation planning can generalize to local editing. Given a zero-one image mask M, we redesign the image editing function as $I_{\text{out}} = o(I, \alpha) \odot M + I \odot (1 - M)$, where $\odot$ is element-wise product; thus only the masked part is edited. Given K mask candidates, we can add an inner loop over all K mask candidates to further generate K edited images each time. In this case, the time complexity goes to $O(NB|\mathcal{O}|K)$. However, K can be removed if we know the grounded mask for each operation. Its full algorithm is described in the Alg. 2. We use UPSNet [14] to obtain the mask candidates and use [6] for removing/inpainting operation. Given each operation with its region, it could also train our T2ONet augmented with the grounding model. Since this paper focuses on global operation planning, it will be left for future work.

D. Time Analysis

We compare the running time of T2ONet and Pix2pixAug [11] in Tab. 7. For T2ONet, the computing-intensive planning is a pre-processing step and only needs to be computed once, and our model shows faster train and test speed than Pix2pixAug, indicating that our method not only has better editing quality, but also is computational cheaper.

Algorithm 2: Operation Planning with Local Editing

Input: I_0, I_g, max operation step N, threshold ϵ, beam size B, operation set $\mathcal{O}$, mask set $\mathcal{M}$

p=[I_0]
$cost(I) = ||I - I_g||_1$
for t *in* $1 : N$ **do**
  $q \leftarrow [\,]$
  for $I \in p$ **do**
    for $o \in \mathcal{O}$ **do**
      for $M \in \mathcal{M}$ **do**
        $\alpha^* = \arg\min_\alpha cost(o(I, \alpha) \odot M + I \odot (1 - M))$
        $I^* \leftarrow o(I, \alpha^*) \odot M + I \odot (1 - M)$
        $q \leftarrow q \cup I^*$
      end
    end
  end
  $q \leftarrow Sort(q), sortkey = cost(I^*)$
  $p = q[: B]$
  for $I \in p$ **do**
    if $cost(I) < \epsilon$ **then**
      Break All Loop
    end
  end
end
$\{o_t\}, \{\alpha_t\}, \{M_t\}, \{I_t\} \leftarrow Backtracking(p)$
return $\{o_t\}, \{\alpha_t\}, \{M_t\}, \{I_t\}$

Method	Planning (s)	Train (s)	Test (s)
Pix2pixAug [11]	18.58	0.37	0.05
T2ONet	-	1.16	0.16

Table 7. The average running time comparison for GAN-based method Pix2pixAug [11] and our method. The training time is computed in batch size 64, and test and planning time are computed in batch size 1.

E. More advantages of T2ONet

E.1. Resolution independent editing

Our model will conduct resolution-independent editing and can produce the output with the same resolution as the input image. However, the GAN-based method suffers from generating high-resolution images, such comparison is shown in Fig. 3.

E.2. Inference with multiple possible output

We have discussed the trade-off between L1 and variance by sampling the operation parameter during training stage, and such variance is measured over the outputs edited from different requests, with the purpose of indicating the

Figure 2. Visualization for ablation study methods. T2ONet(-L1) is the modified version without L1 loss, T2ONet(-L1+D) is to replace the L1 loss to adversarial loss.

Figure 3. Compared with the GAN-based method GeNeVa and Pix2pixAug, although all the methods conduct the correct editing, our method has no pixel distortion and is independent to image resolution.

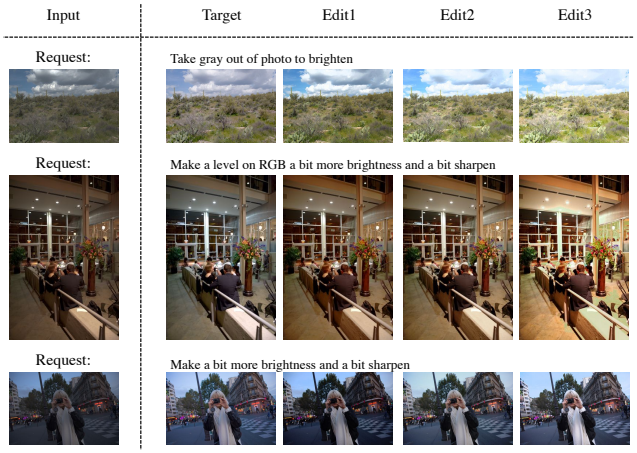

Figure 4. Visualization for diversified output given the same input and request by sampling the operation parameter at inference stage.

model's language-sensitivity However, our model can even generate multiple output given the same request by sampling the operation parameter at the inference stage, whose result is shown in Fig. 4.

F. More visual results

F.1. Comparison Methods

Here we show the comparison visual result of Bilinear-GAN, TAGAN and ManiGAN in Fig 5. The visual results for ManiGAN is quite blur, and its L1, SSIM, FID are 0.1398, 0.5177, 157.4145 on MA5k-Req and 0.1834, 0.4938, 234.6784 on GIER, respectively. Therefore we did not do user study for this method.

F.2. T2ONet

More visual results for T2ONet on MA5k-Req and GIER are shown in Fig. 6 and Fig. 7, respectively.

F.3. Operation Planning

More visualization of the operation editing process is shown in Fig. 8

G. More Experiment Implementation Details

Training images are resized to 128×128, and test/val images are resized to short edge 600 with aspect ratio unchanged. The pixel value is normalized to 0-1

For T2ONet, ResNet18 [3] is used to encoding image into a 512-d feature. The word and operation embedding is 300-d, and the word embedding is initialized by GloVe. Two-layer bi-LSTM with feature with hidden size 256 is used to encode the language request, and two-layer LSTM

Figure 5. The comparison results for BilinearGAN, TAGAN and ManiGAN on MA5k-Req (left) and GIER (right) datasets

decoder has hidden size 512. All the other FC layers output with a 512-d feature.

For operation planning, we adopt Nelder-Mead [7] for parameter optimization. And, for lanugage-guided image editing, the training is alternatively in two losses. For odd iteration, we only optimize $\mathcal{L}_o$ and $\mathcal{L}_\alpha$ in a teacher forcing fashion. For even iteration, we only optimize $\mathcal{L}_{L1}$ using the previously generated action and image as the input for the next state. We take the top-1 operation with its parameters every step. The final image-level $\mathcal{L}_{L1}$ can backward propagate gradients to the weights of T2OCell other than the weights of the FC layer for prediction of the operation o. Hence, in all ablation study of the T2ONet, we always need the loss of $\mathcal{L}_o$ to supervise the operation selection. The model is trained on a single GPU with a 64 batch size.

H. Operation Implementation Details

We adopt six operations: `brightness`, `saturation`, `contrast`, `sharpness`, `tone`, and `color`. The operation modular network is composed of these operations in a fixed order if they are needed. With the input image I, parameter p, and output image I', the implementation of operation submodules are illustrated as follows.

H.1. Brightness and Saturation

The hue, saturation, value in the HSV space of image I are denoted as $H(I)$, $S(I)$, $V(I)$. Here p is an unbounded scalar. Let $V'(I) = \text{clip}((1+p)\cdot V(I), 0, 1)$ and $S'(I) = \text{clip}((1+p)\cdot S(I), 0, 1)$, the output image for brightness operation is

$$I' = \text{HSVtoRGB}(H(I), S(I), V'(I)), \tag{9}$$

and the output image for saturation operation is

$$I' = \text{HSVtoRGB}(H(I), S'(I), V(I)). \tag{10}$$

The HSVtoRGB is a differentiable function mapping the RGB space to HSV space, implemented via Kornia [8], and $\text{clip}(x, 0, 1)$ is a clip function to clip x within 0 to 1.

H.2. Contrast

Contrast operation is controlled by a scalar parameter p, implemented following [4]. First compute the luminance of image I as

$$\text{Lum}(I) = 0.27I_r + 0.67I_g + 0.06I_b, \tag{11}$$

where I_r, I_g, I_b are the RGB channels of I. The enhanced luminance is

$$\text{EnhancedLum}(I) = \frac{1}{2}(1 - \cos(\pi \cdot \text{Lum}(I))), \tag{12}$$

and the image with enhanced contrast is

$$\text{EnhancedC}(I) = I \cdot \frac{\text{EnhancedLum}(I)}{\text{lum}(I)}. \tag{13}$$

The output image I' is the combination of the enhanced contrast and original image

$$I' = (1-p)\cdot I + p \cdot \text{EnhancedC}(I). \tag{14}$$

H.3. Sharpness

The sharpness operation is implemented by adding to the image with its second-order spatial gradient [1], expressed as

$$I' = I + p\Delta^2 I, \tag{15}$$

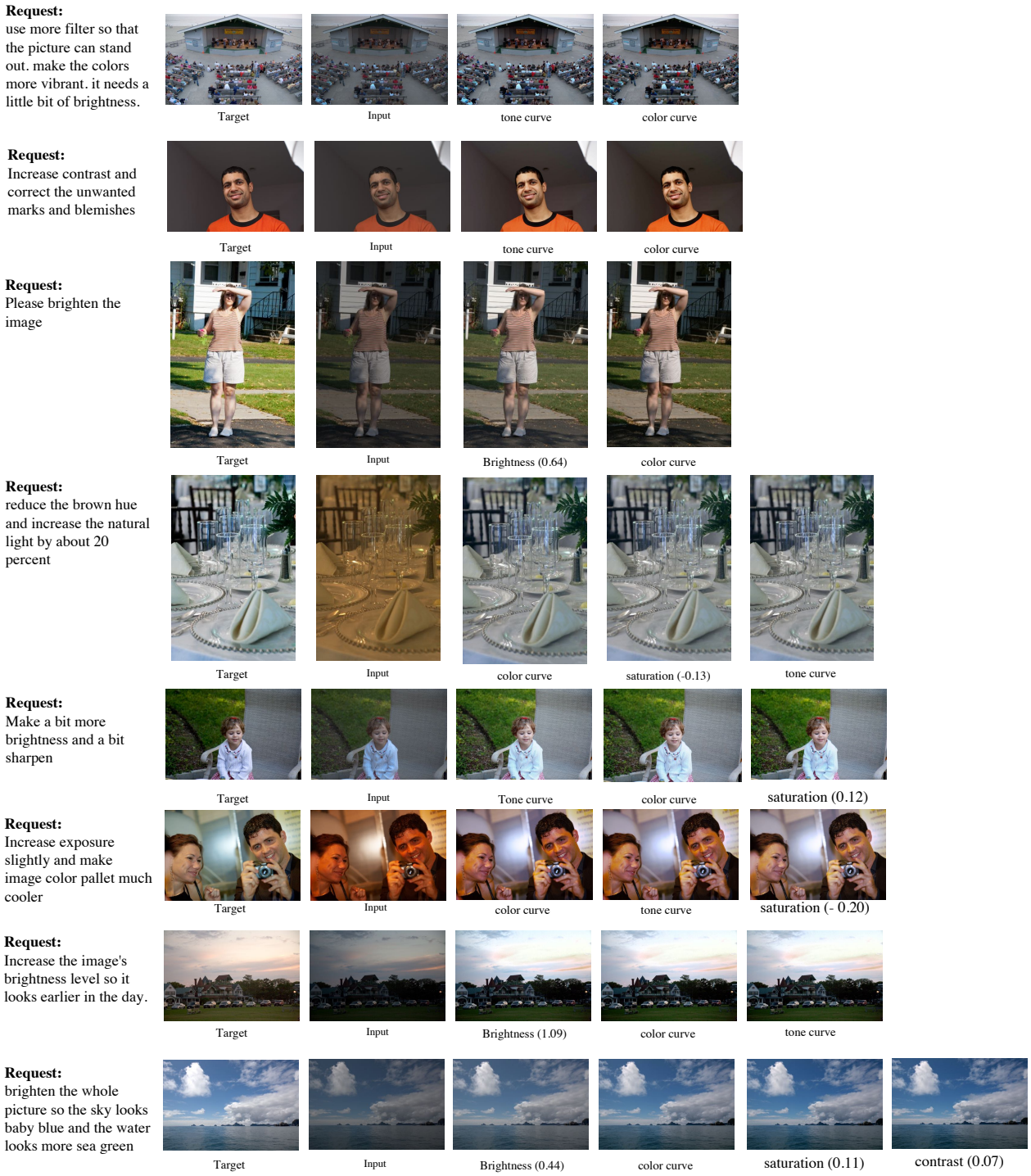

Figure 6. The visual results for T2ONet on MA5k-Req dataset.

where p is a scalar parameter and $(\Delta^2 \cdot)$ is the Laplace operator over the spatial domain of the image. The Laplace operator is applied to each channel of the image.

H.4. Tone and Color

The tone and color operation follows curve representation [4]. The curve is estimated as piece-wise linear functions with N pieces. The parameter $p = \{p_i\}_{i=0}^{M-1}$ is a vector of length M. With the input pixel $x \in [0, 1]$, the output

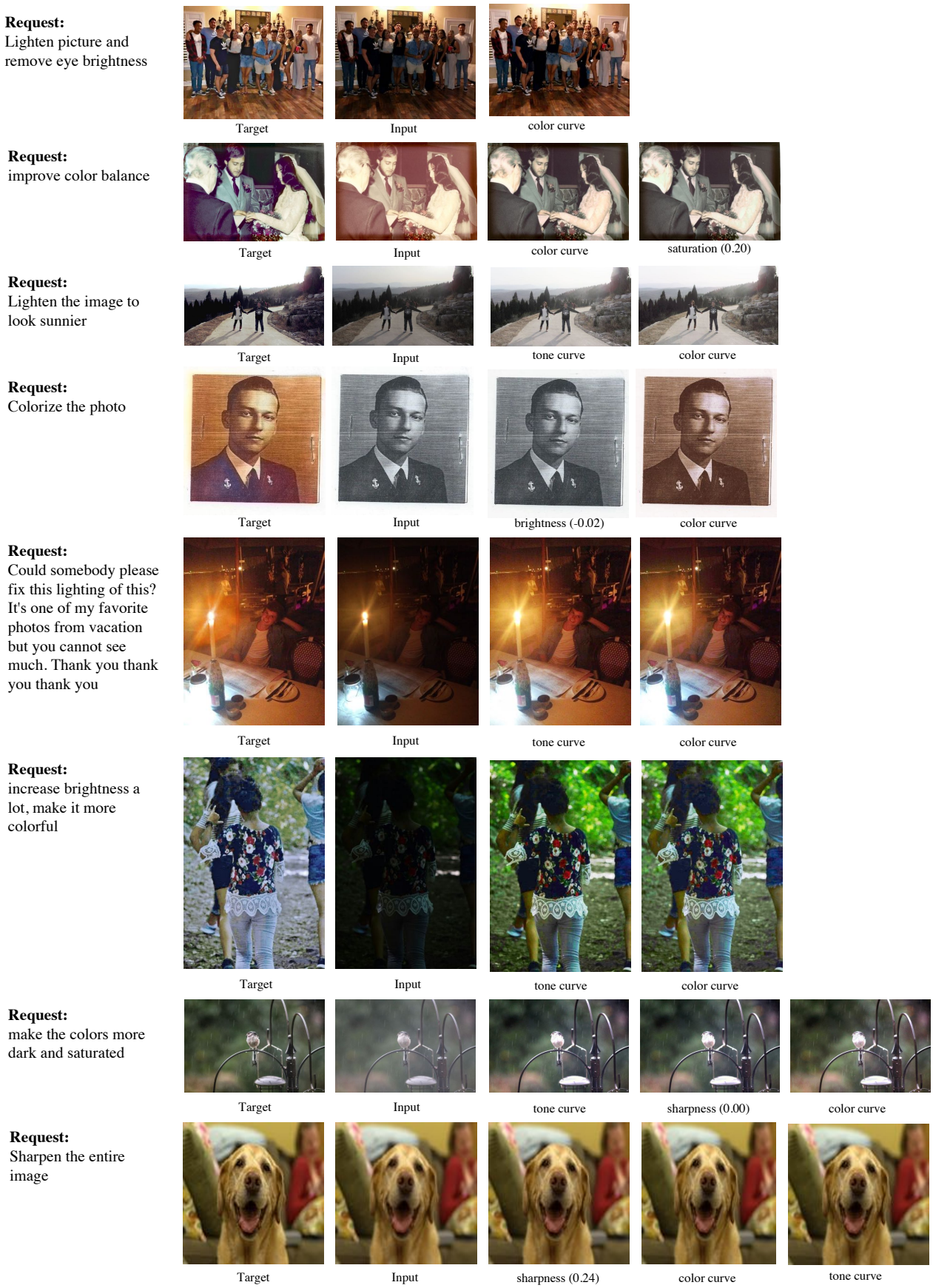

Figure 7. The visual results for T2ONet on GEIR dataset.

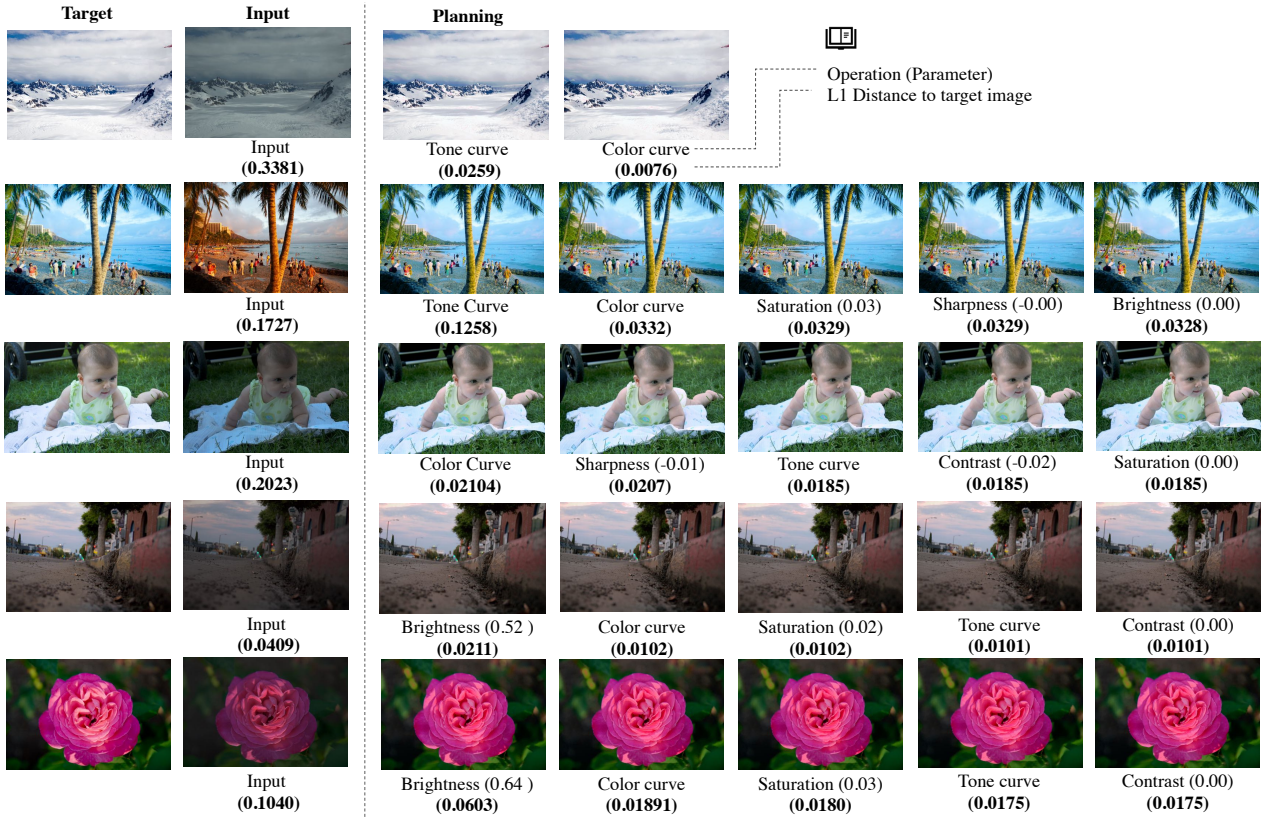

Figure 8. The visual results for operation planning.

pixel intensity is

$$f(x) = \frac{1}{Z} \sum_{i=0}^{N-1} \text{clip}(Nx - i, 0, 1) p_i, \tag{16}$$

where $Z = \sum_{i=0}^{N-1} p_i$. For tone operation, $N = M = 8$, the same f will apply to each of the RGB channels of the image I. For color operation, three different f are applied individually to each of RGB channels. Each $f(x)$ has $N = 8$, leading to $M = 3N = 24$.

I. Languages for Image Variance Evaluation

The 10 different requests are as follows:

1. Decrease the brightness.
2. Increase the brightness.
3. Enhance the color.
4. Decrease the color.
5. Improve contrast.
6. Reduce contrast.
7. Increase saturation.
8. Reduce saturation.
9. Increase the brightness a little.
10. Increase the brightness a lot.

J. Dataset

J.1. More Detail of MA5k-Req Collection Process

In this section we We show the worker the input and target images, and let workers write the editing request. We deploy the annotation collection interface on Amazon Mechanic Turk involving totally 268 workers for FiveK. Each request annotation is 0.03, and we have the approvals for crowdsourcing.

We show the worker the input and target images, and let workers write the editing request. We deploy the annotation collection interface on Amazon Mechanic Turk involving totally 268 workers for FiveK and 197 workers for web images. Each request annotation is $0.03.

For quality control, we initially collect the language requests for a subset of image pairs, and manually select good workers depending on the annotation quality. Then we only allow good workers to annotate the full dataset.

J.2. Visualization of Dataset Samples

Some samples draw from MA5k-Req and GIER is shown in Fig. 9

Figure 9. Data examples draw from MA5k-Req (left) and GIER (right).

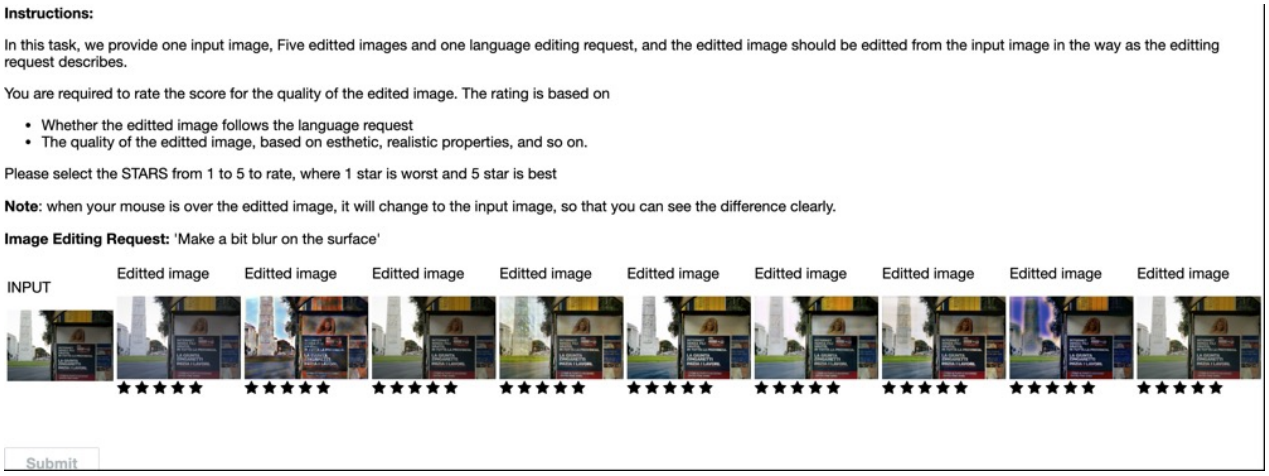

Figure 10. The interface for user study. The edited result of all the methods are shown in random order. The worker should select the star under each edit to indicate the score they rate.

K. User study details

The interface of the user study is shown in Fig. 10.

References

[1] Rafael C Gonzales and Richard E Woods. Digital image processing, 2002. 6

[2] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[4] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018. 2, 3, 6, 7

[5] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 1

[6] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019. 4

[7] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 6

[8] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020. 6

[9] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014. 3

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2

[11] Hai Wang, Jason D Williams, and SingBing Kang. Learning to globally edit images with textual description. *arXiv preprint arXiv:1810.05786*, 2018. 4

[12] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 1

[13] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 3

[14] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, pages 8818–8826, 2019. 4

[15] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018. 4