

## A. Details of Data Preparation

In this section, we detail our data preparation process for training both the policy model and the reward model. We also provide examples of the data.

### A.1. Training Data for Policy Model

**Initial Data.** We start by collecting initial data from user editing histories in Lightroom. Specifically, in a single editing process, a user first inputs an image. Then, the user explores the Lightroom interface, experimenting with different editing operations to achieve their desired result. This process ends with a final image along with the sequence of edits the user performed. In this data collection, we record the *final* result image and the corresponding editing process. The editing process consists of parameterized editing tool-use operations that specify which Lightroom tools were used to obtain the final image, e.g., `{exposure=+0.9; contrast=-30}`.

**Annotating Instruction and Reasoning.** In the collected data above, two essential elements are missing: ❶ the specific editing instruction the user had in mind when modifying the image, and ❷ the reasoning process that guided the user’s use of Lightroom tools during editing. We leverage an MLLM to annotate both the instruction and the reasoning.

We annotate the instruction and reasoning sequentially. First, given a triplet consisting of the before image, the after image, and the parameterized editing operations, the MLLM is asked to infer the user’s editing intention. In this stage, it is essential to produce diverse categories of instructions that reflect the variability in real users’ inputs, e.g., differences in length, complexity, and whether the instruction precisely describes technical terms like a professional photographer or generally describes a desired style like a typical user. To this end, for each input image, we specify different prompts and perform multiple rounds of sampling, resulting in 9 variants of instructions. In the second annotation stage, the MLLM is provided with the same triplet, along with the annotated editing instruction. The goal is to reconstruct the reasoning process the user likely followed when using these tools. Importantly, the reasoning process must be consistent with the final parameterized editing operations. In other words, the reasoning must naturally lead to the actual editing operations applied by the user. We carefully designed the prompts used for these two annotation processes, and the detailed prompts are reported in Appendix F.1.

**Filtering.** To ensure the quality of the collected data, we introduce two filtering stages. ❶ In the first filtering stage, before annotating the instruction and reasoning, we first verify that the user’s edits lead to a visible change between the before and after images. To ensure this, we compute the PSNR value between the two images and filter out image pairs with a PSNR larger than 35 dB, indicating that the images are too similar and that no meaningful editing has actually occurred. ❷ The second filtering stage occurs after the annotation step. Because both the instruction and the reasoning are annotated using MLLMs, they may be inconsistent with the editing configurations that were actually applied. To address this, we use an MLLM to check whether the annotated instruction and reasoning are consistent with the differences between the before and after images as well as with the parameterized editing operations. We then filter out the samples that are found to be inconsistent.

After these steps, we obtain a dataset of 190k examples for policy model training. Each example consists of a before-editing image, a user-edited after image, the user-performed editing operations, and the annotated user instruction and reasoning process, all carefully synthesized and filtered by the MLLM.

### A.2. Training Data for Reward Model

**Initial Data.** To train the reward model, the training data should contain image retouching results with diverse quality levels, and we must ensure that, under the same instruction, a better retouching result receives a higher reward than a suboptimal one. To achieve this, we begin with the user editing dataset similar to the one used for training the policy model, which includes before-and-after image pairs, parameterized editing operations, and the annotated editing instructions. Based on this dataset, our next step is to produce editing variants with different levels of editing quality.

**Prepare Suboptimal Edits.** Since the current after-edited image is consistent with the annotated instruction, perturbing the editing operations will cause the final retouching to deviate from the gold edit, resulting in a *sub-optimal* outcome. Therefore, we design a perturbation baseline, whose core component is an MLLM responsible for generating perturbed editing operations. Specifically, given the current editing operations, the before–after image pair, and the instructions, the

MLLM is asked to *intentionally* modify the editing operations so that the final retouching violates the specified instructions in various ways. For example, a suboptimal perturbation might decrease the temperature when the instruction expresses a desire for a warmer atmosphere, or excessively adjust the exposure to produce a perceptually degraded image. To support diverse perturbations, we sample perturbed edits three times for each image pair. After this step, we leverage Lightroom to render the corresponding perturbed images. As a result, we obtain triplets of three images: the before image, the user-edited after image (strong response), and the perturbed after-edited image (weak response) generated in our perturbation step.

**Prepare Metrics and Rewards.** The next step is to prepare the *metrics* and *rewards* used for supervised fine-tuning of the reward model. For successful SFT, the metrics should consist of a set of textual descriptions that highlight the important aesthetic characteristics of successful edits given the input images and instructions. The reward should then produced analysis the retouching results and assigning a scalar value. We leverage an MLLM to annotate both the metrics and the rewards correspondingly, and the prompts will be detailed in Appendix F.1. To ensure precise annotations, we perform filtering, which will be detailed next.

**Filtering.** We perform a two-stage filtering process to ensure the quality of both the perturbations and the annotations. ❶ For perturbation, we leverage an MLLM to verify whether the perturbed image can indeed be considered a weaker retouching result compared with the user-edited image. The weakness may arise from inferior aesthetics or inconsistencies with the instructions. If the perturbation does *not* make the editing worse, we discard that perturbation. ❷ For annotation, since we have ground-truth knowledge of which edit is stronger, we filter out the annotated metrics and rewards if the final annotated reward for the user-edited image is not higher than that of the perturbed image. Note that we discard both the metrics and the reward in this case, since an incorrect reward may stem from either the metrics annotation or the reward annotation, or both of them.

After these steps, we obtain a dataset of 10k examples for reward model training. Each example consists of: ❶ Images: a before-editing image, a user-edited after image, a perturbed after image, and ❷ Text: the user-performed editing operations, and the annotated metrics and rewards. Besides, we emphasize that in the policy-guided reward training, instead of using the perturbed suboptimal images, we leverage the model-generated images, resulting in an additional 5k training examples for reinforcement learning.

We present training data examples for policy model and reward model in the next page.

### Training Data Example (Policy Model):

#### Model Input:

You are a Lightroom adjustment expert. Convert a user's image edit description into Lightroom parameter values. First reason (<think>), then output (<answer>) as valid Lua.

#### Rules:

1. In <think>: comprehensively infer needed Lightroom changes.
2. In <answer>: output 'return ... ' with only changed params.

#### Schema:

Temperature: 2000 to 10000, lower values cooler (blue), higher values warmer (yellow), default is 4500-6500

Tint: -150 to +150

Exposure2012: -5.0 to +5.0

Contrast2012, Highlights2012, Shadows2012, Whites2012, Blacks2012: -100 to +100

... (Detailed tools considered are introduced in [Appendix F.1](#))

#### Instructions:

Make the cherry blossoms pop more, creating a sense of fresh morning light drifting across the petals, letting the moment feel calm, delicate, and touched by a quiet, hopeful energy.

#### Input Image:



#### Model Output:

<think>First, I'll increase the overall brightness to create a more inviting atmosphere, letting the scene feel immediately lighter and more open. Next, I'll deepen the sense of presence in the cherry blossoms by strengthening their definition; adding a noticeable boost in clarity, texture, and contrast so the petals feel crisp and vivid without appearing harsh. I'll warm the image slightly while keeping a hint of coolness in the tint, giving the light a natural spring balance. A touch of vibrance and saturation will help the blossoms feel livelier, while a bit of dehaze brings the entire view into clearer focus. Altogether, these choices create a bright, fresh moment where the blossoms glow gently against the garden beyond. </think>

<answer>return

```
{Blacks2012 = 5, Clarity2012 = 48, Contrast2012 = 42, Dehaze = 50, Exposure2012 = 1.19, Highlights2012 = 17, IncrementalTemperature = 19, IncrementalTint = -3, Saturation = 10, Shadows2012 = -1, Texture = 17, ... , ToneCurveName2012 = `Custom`, ToneCurvePV2012 = [1] = 0, [2] = 3, [3] = 63, [4] = 52, [5] = 255, [6] = 255, , Vibrance = 22, WhiteBalance = `Custom`, Whites2012 = 14}
```

#### Output Image Rendered by Lightroom:



### Training Data Example (Reward Model):

#### Model Input:

Given a before image, an instruction, and two after-edited images (one stronger, one weaker), set up a series of principles and analytically identify why one edit is stronger and the other is weaker.

The user instruction is: I wanted the scene to feel more vibrant and dynamic. The colors pop more, especially the reds and blues, making the image look lively and engaging. The contrast and clarity adjustments enhance the details, bringing out the texture in the grass and the lamp's details.

In the following images, the first one is the before image. You will evaluate the rest two after-edited images.



#### Model Output:

Specific Criteria:

Criteria 1 (Weight: 25%): Vibrancy of Key Colors - The edited image should enhance the saturation of reds and blues to make them pop, contributing to a lively and engaging visual feel.

Criteria 2 (Weight: 20%): Contrast Enhancement - The image should use increased contrast to bring out details in elements like grass and the lamp, rather than reducing contrast.

Criteria 3 (Weight: 20%): Detail Preservation and Enhancement - Clarity adjustments should enhance texture in the grass and lamp details, avoiding excessive noise reduction or softening that loses detail.

Criteria 4 (Weight: 15%): Dynamic Range and Tonal Balance - The image should maintain good dynamic range with a balanced tone curve, preventing flattening or over-darkening/lightening.

Criteria 5 (Weight: 20%): Absence of Irrelevant Effects - The edit should not include unnecessary adjustments (e.g., split toning, excessive noise reduction) that distract from the vibrant, dynamic goal.

Analysis:

**\*\*Image 2\*\*** (first edited image):

- **\*\*Criteria 1\*\***: The reds and blues are muted and lack vibrancy, failing to create a lively feel. Score: 3/10.
- **\*\*Criteria 2\*\***: Contrast is reduced—grassy areas appear flat, and lamp details are lost. Score: 2/10.
- **\*\*Criteria 3\*\***: Excessive noise reduction softens grass texture and lamp details, reducing clarity. Score: 2/10.
- **\*\*Criteria 4\*\***: The tone curve flattens the image (e.g., gray sky, washed-out tones), resulting in poor dynamic range. Score: 3/10.
- **\*\*Criteria 5\*\***: Irrelevant adjustments (e.g., muted colors, flattened tones) distract from the vibrant goal. Score: 2/10.

**\*\*Image 3\*\*** (second edited image):

- **\*\*Criteria 1\*\***: Reds and blues are vibrant (e.g., bright blue sky, lively greens), creating a dynamic feel. Score: 9/10.
- **\*\*Criteria 2\*\***: Contrast is enhanced—grass has depth, and lamp details (e.g., metal texture) are sharp. Score: 9/10.
- **\*\*Criteria 3\*\***: Clarity adjustments preserve grass texture and lamp details, maintaining sharpness. Score: 9/10.
- **\*\*Criteria 4\*\***: Good dynamic range—sky is bright but not overexposed, and grass has natural shadows/highlights. Score: 8/10.
- **\*\*Criteria 5\*\***: No irrelevant effects; colors and tones align with the vibrant, dynamic goal. Score: 9/10.

Scores:

**\*\*Image 2\*\***:  $(3 \times 0.25) + (2 \times 0.20) + (2 \times 0.20) + (3 \times 0.15) + (2 \times 0.20) = 0.75 + 0.4 + 0.4 + 0.45 + 0.4 = 2.4$

**\*\*Image 3\*\***:  $(9 \times 0.25) + (9 \times 0.20) + (9 \times 0.20) + (8 \times 0.15) + (9 \times 0.20) = 2.25 + 1.8 + 1.8 + 1.2 + 1.8 = 8.85$

Final Evaluation:

2.4, 8.85

	Value		Value	
<b>SFT</b>	Learning rate (LLM)	2e-7	Learning rate (LLM)	2e-7
	Learning rate (Projector)	1e-5	Learning rate (Projector)	1e-5
	Learning rate (Vision tower)	Freeze	Learning rate (Vision tower)	Freeze
	Scheduler	Cosine	Scheduler	Cosine
	Training data	10k	<b>SFT</b> Training data	10k
	Model max length	8192	Model max length	8192
	Batch size	32	Batch size	32
	Image max pixel	512 × 28 × 28	Image max pixel	512 × 28 × 28
	Image min pixel	256 × 28 × 28	Image min pixel	256 × 28 × 28
	Deepspeed optimization	zero3	Deepspeed optimization	zero3
Pretrained Checkpoint	Qwen-2.5-VL-7B	Pretrained Checkpoint	Qwen-2.5-VL-7B	
<b>RL</b>	GRPO batch size	64	GRPO batch size	64
	Max prompt length	1024	Max prompt length	2048
	Max response length	4096	Max response length	2048
	Use KL Loss	False	<b>RL</b> Use KL Loss	False
	Rollout temperature	0.8	Rollout temperature	0.8
	Rollout responses	5	Rollout responses	8
	DAPO overlong buffer size	1024	DAPO overlong buffer size	1024
	Learning rate	1e-6	Learning rate	1e-6

Table 3. Hyperparameters in Training. **Left:** Hyperparameters for the policy model. **Right:** Hyperparameters for the reward model.

## B. Details of Implementation

In this section, we describe the training details for both the policy model and the reward model. The detailed hyperparameters are listed in Table 3.

**Policy Model.** The policy model is based on Qwen-2.5-VL-7B, and we use the officially released checkpoints. The full 190k training dataset is divided into two parts: 180k samples for supervised fine-tuning (SFT) and 10k samples reserved for reinforcement learning (RL). During the SFT stage, we use the official QwenLM repository as the training codebase. We set the maximum sequence length to 8192 tokens. All other hyperparameters follow the defaults in the original repository, and we list them in Table 3. SFT is conducted on 8 A100 GPUs. For the reinforcement learning stage, we use the verl repository as the training codebase. We adopt the DAPO variant of GRPO training with a batch size of 64, a maximum prompt length of 1024, and a maximum response length of 4096. Additional hyperparameters are reported in Table 3. RL training uses a total of 24 A100 GPUs: 8 GPUs for policy rollout and updates, and 16 GPUs for hosting four online reward model instances. To support RL with visual feedback, we also deploy a dedicated Lightroom server to render the output images. For each RL update, the policy-generated responses are first sent to the Lightroom server to obtain the rendered images. The input image, output image, and instruction are then forwarded to the four reward model servers, which compute the rewards and return them to the policy model.

**Reward Model.** The reward model is also built on Qwen-2.5-VL-7B. Similar to the policy model, we use the QwenLM and verl repositories for the SFT and RL phases, respectively. In the initial stage, the reward model is trained only on the perturbed 10k dataset, where 8k examples are used for SFT and the remaining 2k are used for RL. In a subsequent policy-guided training stage, we further incorporate 5k additional samples generated by the policy model to strengthen the reward model through continued RL updates. The full set of hyperparameters used for reward-model training is presented in Table 3.

## C. Human Study

In this section, we conduct a human study to further evaluate the performance of RETOUCHIQ and the baselines. The study is carried out on Amazon Mechanical Turk. We sample 25 instruction-image pairs covering categories of quality improvement, style adjustment, and local retouching. Each question is awarded \$0.03. For each question, we show participants the edited image produced by one method, together with the input image and the instruction, and ask them to answer two evaluation questions: ① Semantic Consistency: How well does the final edited image follow the user instruction? ② Perceptual Quality: How is the overall perceptual quality of the final edited image? Participants rate each question on

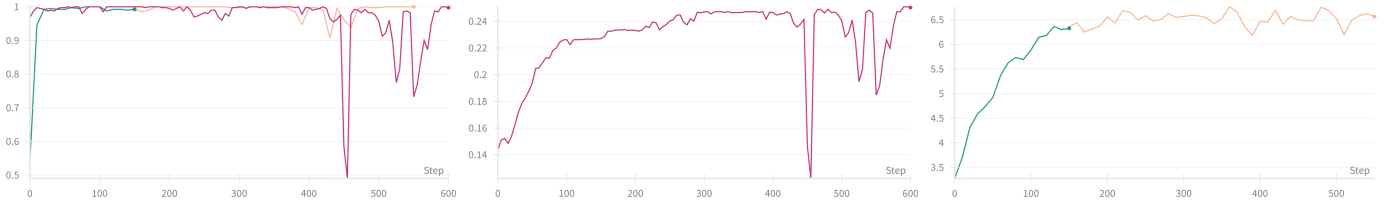


Figure 7. Testing reward wandb curves under different reward guidance settings. **Pink curve:** Using the verifiable reward. **Green curve:** Initial training of the policy model, where the reward model is trained on perturbed data. **Orange curve:** Continued training of the policy model, where the reward model is further fine-tuned with PGRT. **Left:** Format reward. **Middle:** Verifiable reward following [22]. **Right:** Reward-model score. Note that the discontinuity in the reward-model curve is caused by the reward model shifting during PGRT.

a scale from 0 to 2. To ensure reliable judgments, we provide detailed explanations, a scoring rubric, and illustrative examples.

Figure 9 shows an example of the interface that participants will see during the human study. We report the human study results in Table 4. The results are consistent with our findings from the qualitative analysis. Specifically, the general-purpose MLLMs fail to generate high-quality retouched images, and MONET-GPT has difficulties following customized instructions. Our method achieves comparable semantic consistency to JARVISART and demonstrates strong perceptual quality.

Model	Consistency	Quality
GPT-5	0.57	0.40
GEMINI-2.5	0.41	0.46
MONETGPT	0.53	0.63
JARVISART	0.73	0.70
RETOUCHIQ	0.73	0.77

Table 4. Human study results. We report the semantic consistency and perceptual quality rated by human.

## D. Training Curve Analysis

In this section, we present and compare training variants of the policy model through testing reward curves, shown in Figure 7. We evaluate three variants of the policy model: ❶ policy model trained using verifiable reward following prior work [22] (pink curve); ❷ policy model trained with reward model guidance, where the reward model is trained using perturbed suboptimal edits as described in Sec. 4.2 and Appendix A.2 (green curve); and ❸ policy model trained with reward model guidance, where the reward model is trained with PGRT (orange curve). We present three curves. The **left** plot corresponds to the format reward, which checks whether the policy model first outputs its reasoning and then produces well-formatted parameterized configurations that can be successfully parsed by Lightroom. The **middle** plot corresponds to the verifiable reward. The **right** plot shows the score provided by the reward model. We observe the following:

**Unstable training with verifiable reward.** We observe that the verifiable reward tends to provide unstable guidance, as indicated by the pink curves in the first two plots (e.g., at steps 450, 520, and 540). We hypothesize that this instability arises because the verifiable reward computes color-distribution differences using single reference images, which can lead to noisy reward signals and degrade the model, sometimes even resulting in underperforming checkpoints with incorrect output formats. In contrast, the reward model provides relatively stable guidance, leading to a smooth improvement in model performance, as shown in the right plot.

**Reward model score improvements during two-stage RL training.** From the right plot, we observe a clear improvement in the model’s retouching quality over the course of training. Initially, the reward score is around 3, indicating that the retouching results deviate substantially from those produced by the supervised fine-tuning checkpoints. The reward score gradually increases throughout both training stages. Notably, after switching to the reward model trained with PGRT, the policy model’s score further improves. This demonstrates that PGRT provides more accurate and reliable reward signals, enabling the policy model to better align with high-quality retouching objectives and achieve more consistent improvements.

**Format reward.** Finally, we observe that regardless of whether verifiable reward or reward-model guidance is used, the model quickly learns the correct output format and produces parsable parameterized editing operations, as shown in the left plot. This is consistent with prior findings showing that format reward is learned rapidly during RL training.

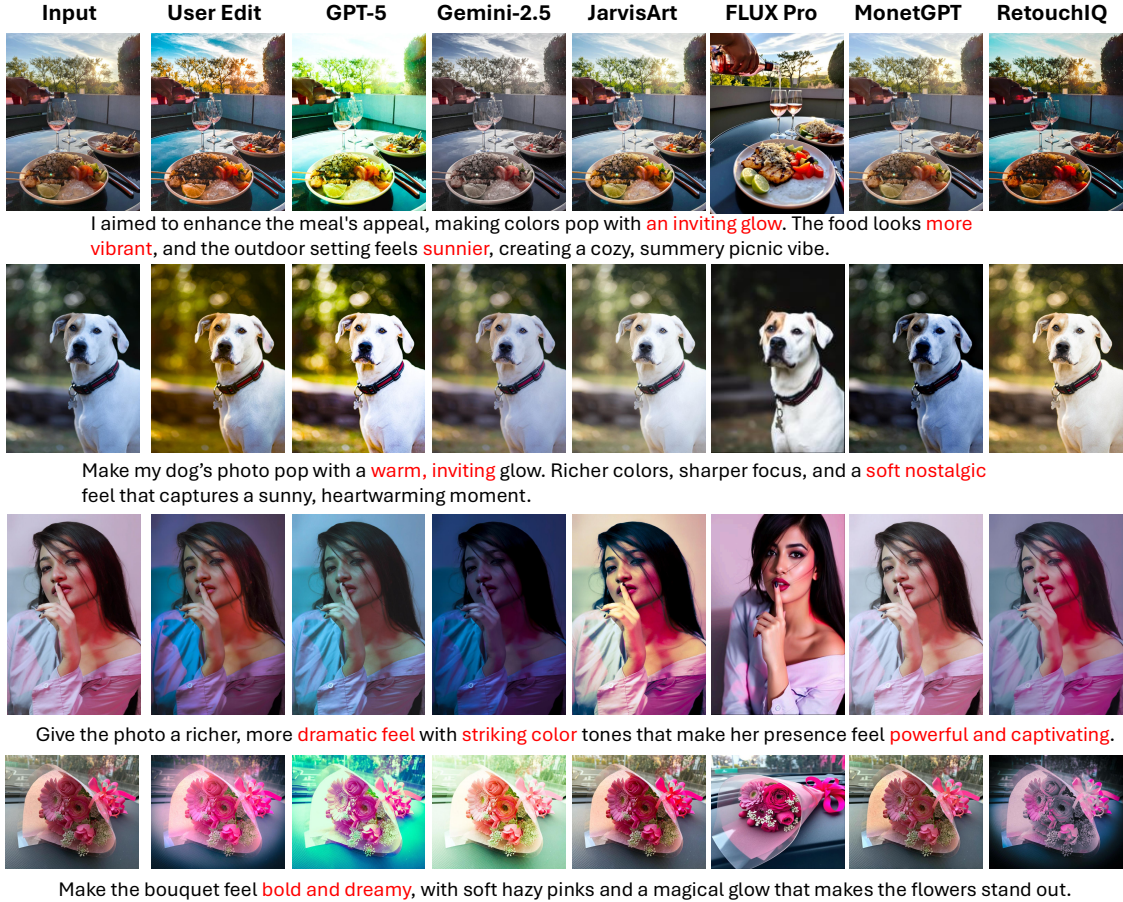


Figure 8. Qualitative results across diverse image retouching scenarios, including quality enhancement (top), style transformation (middle), and local retouching (bottom). For each example, the input image and the corresponding user-edited result are shown on the left.

## E. More Visual Examples

In this section, we provide additional visual examples of our retouching results alongside those of the baselines. These examples are shown in Figure 8. We highlight the following observations:

**Failure modes of baselines.** Consistent with the analysis in Sec. 5.2, we observe that different categories of baselines exhibit distinct failure patterns in the retouching task. ① General-purpose MLLMs such as GPT-5 and Gemini-2.5 often introduce drastic and unintended changes to image attributes, e.g., colors, shadows, or global tones in the first, third, and fourth rows. This likely stems from the fact that these models must infer natural parameter configurations in a purely zero-shot manner and lack knowledge of realistic edit magnitudes. ② Diffusion-based models like Flux Pro frequently alter the image content itself, deviating from the original scene. ③ MonetGPT struggles to follow user instructions and sometimes produces results that are inconsistent with the given prompts, as seen in the second row.

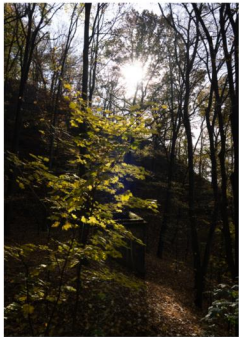
**Various plausible edits.** We also observe that for certain high-level or general instructions, multiple distinct editing styles can reasonably satisfy the user’s intent. For example, in the fourth row, the instruction requests a “bold and dreamy” result. Both JarvisArt and RetouchIQ produce edits that meet this description, yet they do so through different stylistic choices, each offering a compelling portrait from its own perspective. This variability is an intrinsic characteristic of the image-retouching task, where user intent may permit multiple valid interpretations. At the same time, we notice that MLLM-based judges sometimes exhibit a preference for specific editing styles. For instance, in this example, the MLLM assigns higher semantic-consistency scores to JarvisArt (e.g., 8 vs. 4 in the last row). Given the subjectivity of these instructions, there is a potential need to combine automatic evaluation methods (e.g., MLLM-as-a-judge) with qualitative visual inspection to gain a more balanced understanding of model performance.

**Instructions:** Your task is to evaluate the effectiveness of an edited image based on the provided input image and the given instruction.

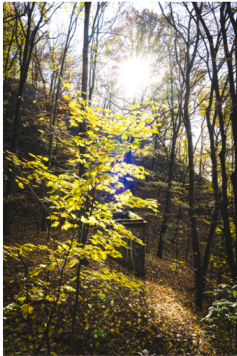
Rate how well the edited image aligns with the instruction on a scale from 0 to 2:

- 0: The edited image does not match the instruction at all. The visual changes differ significantly from the intended purpose described in the instruction.
- 1: The edited image partially reflects the instruction. It includes some intended changes, but may also contain irrelevant modifications or may miss important changes.
- 2: The edited image fully satisfies the instruction. All visual changes align with the intended purpose without introducing unnecessary or incorrect modifications.

The original image is:



Question: Evaluate how **consistent** the edited image is with the instruction: "I wanted the forest to glow warmly, bringing out the golden leaves while keeping the mood serene."



- 0
- 1
- 2

**Submit**

Figure 9. The example interface of Amazon Mechanical Turk in our human study.

## F. Prompts

In this section, we provide the prompts used for data construction, baseline inference, and metric evaluation.

### F.1. Prompts for Data Construction

#### **Prompts for Instruction Annotation (Variant 1 - for long instruction):**

You are given three inputs:

1. The original image (before editing)
2. The edited image (after editing)
3. The editing configuration parameters

Your goal is to infer the user’s editing intention — what they were trying to achieve visually or emotionally — and express it as if the user is describing what they want before editing.

Silently compare the two images and settings to understand: How visual qualities changed (lighting, tone, color, clarity, subject focus).

What emotional or atmospheric impression the result conveys (e.g., calm, nostalgic, lively, intimate, surreal).

Then write one natural, first-person sentence capturing what the user likely wanted.

Output rules:

Write in first-person voice, but use varied, natural openings. For example, you may begin with: “Please make it . . .”, “Let’s make it . . .”, “It should feel . . .”, “Can you give it . . .”, or other natural phrasing.

Describe how the scene should feel or what atmosphere it should evoke, not its technical features.

Use specific sensory or emotional cues (light, time of day, movement, temperature, softness, energy).

Avoid overused generic adjectives like dramatic, moody, cinematic, vintage, vibrant — use them only rarely, when necessary, and always paired with a specific sensory detail (e.g., “cinematic, like a quiet evening,” “vintage with soft golden warmth”).

Do not mention “before,” “after,” “edit,” or “image.”

Keep it concise: one sentence, around 40 words.

Examples:

1. Let’s make it feel calm and sunlit, with a gentle atmosphere, natural tones, and the quiet warmth of an afternoon where everything slows down and feels soft and peaceful.
2. I’d love it if this felt more alive — a touch brighter, more open, with airy light and a sense of quiet optimism running through the scene.
3. It should feel tender and nostalgic, with softened light, warm colors, and textures that carry the mood of a fleeting, familiar moment that lingers just a little longer.

Last step: recheck two things before finalizing:

1. Overuse of generic adjectives — words like dramatic, cinematic, vintage, moody, and vibrant should be used sparingly and always paired with a specific sensory detail.
2. Use varied openings to keep tone natural and diverse.

Return only the final user instruction sentence.

*(Two input images provided here)*

The configuration details are as follows:

```
{Blacks2012 = 5, Clarity2012 = 48, Contrast2012 = 42, Dehaze = 50, Exposure2012 = 1.19, Highlights2012 = 17, IncrementalTemperature = 19, IncrementalTint = -3, Saturation = 10, Shadows2012 = -1, Texture = 17, ...(Omit) , ToneCurveName2012 = `Custom`, ToneCurvePV2012 = [1] = 0, [2] = 3, [3] = 63, [4] = 52, [5] = 255, [6] = 255, , Vibrance = 22, WhiteBalance = `Custom`, Whites2012 = 14}
```

**Prompts for Instruction Annotation (Variant 2 - for phrase instruction):**

You are given three inputs:

1. The original image (before editing)
2. The edited image (after editing)
3. The editing configuration parameters

Your task is to infer the user's editing intention - the visual or emotional transformation they aimed to achieve - and express it using 3-5 single, meaningful words.

Silently compare the two images and the configuration to understand:

- \* How visual qualities changed (lighting, tone, color, contrast, texture, subject focus).
- \* What overall mood, atmosphere, or stylistic direction the edit conveys (e.g., calm, nostalgic, bright, minimal, warm, surreal).

Output rules:

- \* Output only 3-5 single adjectives or style words, separated by commas (e.g., *warm, natural, soft, calm*).
- \* Words should describe the feeling or visual style, not the tools or parameters used.
- \* Avoid generic terms such as *cinematic, dramatic, moody, vintage, vibrant*. Use them only when truly appropriate and pair them with diverse complementary descriptors.
- \* Prefer specific, grounded, and varied adjectives that convey sensory qualities (light, tone, temperature, color, clarity, emotion).
- \* Do not include phrases, nouns, or technical terms — only adjectives.
- \* Keep the output clean and simple: lowercase, comma-separated, no explanations.

Examples:

1. warm, natural, soft, calm, airy
2. earthy, gentle, muted, timeless
3. cool, minimal, balanced, serene
4. bright, natural, relaxed, authentic
5. pastel, nostalgic, light, tender

Return only the final 3-5 one-word descriptors, separated by commas.

*(Two input images provided here)*

The configuration details are as follows:

```
{Blacks2012 = 5, Clarity2012 = 48, Contrast2012 = 42, Dehaze = 50, Exposure2012 = 1.19, Highlights2012 = 17, IncrementalTemperature = 19, IncrementalTint = -3, Saturation = 10, Shadows2012 = -1, Texture = 17, ...(Omit) , ToneCurveName2012 = `Custom`, ToneCurvePV2012 = [1] = 0, [2] = 3, [3] = 63, [4] = 52, [5] = 255, [6] = 255, , Vibrance = 22, WhiteBalance = `Custom`, Whites2012 = 14}
```

**Prompts for Reasoning Annotation (Adapt from [22]):**

You will be given before-and-after images of user edits, along with the user's instructions for performing those edits. Your goal is to output a thoughtful, step-by-step reasoning process, describing how you would perform the edits starting from the input image and the user's instructions, ultimately arriving at the final result.

Note: You will be provided with the after image, but you must not mention, refer to, or imply its existence. The after image is only for your internal reference to ensure your reasoning is correct. Also, do not mention "before image" — you should act as if you only see one input image and the user's instruction.

To help you generate accurate reasoning, you will also be provided with a configuration file containing the final editing steps. Under no circumstances may you mention, refer to, or imply the existence of this configuration file. You must act as if you only have the input image and the user's instructions, and construct your reasoning accordingly.

Your output thinking process should:

- Present the reasoning as a natural, step-by-step thought process, arriving at the edits in the configuration file through observation and analysis.
- Produce the exact adjustments (values, tools, methods) that match the configuration file. Match the values and tools appearing in the file.

*(Continue in next page)*

## Prompts for Reasoning Annotation (Adapt from [22]):

(Continued)

For your reference, the explanation of the configuration file is provided below:

### Basic Adjustments

- **WhiteBalance:** [As Shot, Auto, Custom] – Controls overall color temperature of the image.
- **Temperature:** [2000–10000] – Blue–yellow color balance; lower values produce cooler (blue) tones, higher values produce warmer (yellow) tones; typical default range is 4500–6500.
- **Tint:** [-150 to +150] – Green–magenta balance; negative values add green, positive values add magenta; typical default range is -50 to +50.
- **Exposure2012:** [-5.0 to +5.0] – Overall brightness of the image (in stops).
- **Contrast2012:** [-100 to +100] – Adjusts contrast between light and dark areas.
- **Highlights2012:** [-100 to +100] – Adjusts bright regions of the image.
- **Shadows2012:** [-100 to +100] – Adjusts dark regions of the image.
- **Whites2012:** [-100 to +100] – Fine-tunes the brightest parts of the image.
- **Blacks2012:** [-100 to +100] – Fine-tunes the darkest parts of the image.
- **Texture:** [-100 to +100] – Enhances or smooths medium-detail textures.
- **Clarity2012:** [-100 to +100] – Controls local contrast, affecting mid-tone edges.
- **Dehaze:** [-100 to +100] – Reduces or adds atmospheric haze.
- **Vibrance:** [-100 to +100] – Increases saturation of less-saturated colors while protecting already saturated colors.
- **Saturation:** [-100 to +100] – Adjusts overall color intensity.
- **IncrementalTemperature:** [-100 to +100] – Increases or decreases color temperature.
- **IncrementalTint:** [-100 to +100] – Increases or decreases color tint.

### Tone Curve

- **ToneCurveName2012:** [Linear, Custom] – Predefined tone curve shapes; default is *Linear*. If `ToneCurvePV2012` or `ToneCurvePV2012Red/Green/Blue` is modified, the setting becomes *Custom*.
- **ToneCurvePV2012:** Custom RGB tone-curve points {[1]=x1, [2]=y1, [3]=x2, [4]=y2, ...}, where x and y range from 0–255; curve is interpolated via `CubicSpline`.
- **ToneCurvePV2012Red/Green/Blue:** Channel-specific tone curves with the same format as above.
- **ParametricShadows/Darks/Lights/Highlights:** [-100 to +100] – Adjust tonal regions individually.
- **ParametricShadowSplit:** [10–50] – Boundary between shadows and darks (default: 25), lower than `ParametricMidtoneSplit`.
- **ParametricMidtoneSplit:** [25–75] – Boundary between darks and lights (default: 50), lower than `ParametricHighlightSplit`.
- **ParametricHighlightSplit:** [50–90] – Boundary between lights and highlights (default: 75).

### Detail

- **Sharpness:** [0–150] – Enhances edge definition.
- **SharpenRadius:** [0.5–3.0] – Width of sharpening along edges.
- **SharpenDetail:** [0–100] – Controls detail-level sharpening.
- **SharpenEdgeMasking:** [0–100] – Masks sharpening to edges.
- **LuminanceSmoothing:** [0–100] – Reduces luminance noise.
- **ColorNoiseReduction:** [0–100] – Reduces chroma noise.
- **ColorNoiseReductionDetail:** [0–100] – Fine-tunes color noise reduction detail.
- **ColorNoiseReductionSmoothness:** [0–100] – Controls smoothness of color noise reduction.

### HSL/Color

- **HueAdjustmentRed/Orange/Yellow/Green/Aqua/Blue/Purple/Magenta:** [-100 to +100] – Shifts hue of each color range.
- **SaturationAdjustmentRed/Orange/Yellow/Green/Aqua/Blue/Purple/Magenta:** [-100 to +100] – Adjusts saturation of each color range.
- **LuminanceAdjustmentRed/Orange/Yellow/Green/Aqua/Blue/Purple/Magenta:** [-100 to +100] – Adjusts brightness of each color range.

### Color Grading

- **SplitToningShadowHue/HighlightHue:** [0–359] – Hue for shadow or highlight toning.
- **SplitToningShadowSaturation/HighlightSaturation:** [0–100] – Saturation for shadow or highlight tones.
- **SplitToningBalance:** [-100 to +100] – Balance between shadow and highlight toning.
- **ColorGradeMidtoneHue/Sat/Lum:** [0–359 for Hue; 0–100 for Sat and Lum] – Adjusts midtone color grade.
- **ColorGradeShadowLum/MidtoneLum/HighlightLum:** [0–100] – Luminance adjustments for tonal regions.
- **ColorGradeBlending:** [0–100] – Blending strength of color grading (default: 50).
- **ColorGradeGlobalHue/Sat/Lum:** [0–359 for Hue; 0–100 for Sat and Lum] – Global grading adjustments.

(Continue in next page)

## Prompts for Reasoning Annotation (Adapt from [22]):

(Continued)

### Effects

- **PostCropVignetteAmount:** [-100 to +100] – Darkens or brightens image corners.
- **GrainAmount:** [0–100] – Adds film grain.
- **ShadowTint:** [-100 to +100] – Adds tint to shadow regions.

### Camera Calibration

- **RedHue/GreenHue/BlueHue:** [-100 to +100] – Shifts hue of primary color channels.
- **RedSaturation/GreenSaturation/BlueSaturation:** [-100 to +100] – Adjusts saturation of primary color channels.

### Lens Blur

The LensBlur parameter simulates depth-of-field effects:

- **Active:** [true, false] – Enables or disables lens blur.
- **BlurAmount:** [0–100] – Strength of blur effect.
- **FocalRange:** “x1 y1 x2 y2” – Defines focal plane and transition zone.
- **BokehShape:** Default is 0.
- **BokehShapeDetail:** [0–100] – Controls edge detail of bokeh shapes.
- **HighlightsThreshold:** [0–100] – Threshold for bright bokeh highlights.
- **HighlightsBoost:** [0–100] – Boosts brightness of out-of-focus highlights.
- **CatEyeAmount:** [0–100] – Simulates cat-eye bokeh near frame edges.
- **CatEyeScale:** [0–100] – Controls size of cat-eye effect.

### Advanced Color Grading (PointColors)

PointColors allows precise control over specific colors. When a parameter value is -1, it indicates the parameter is not used.

- **SrcHue:** [0–6.28] – Source hue in radians (0–360°).
- **SrcSat/SrcLum:** [0–1.0] – Source saturation and luminance.
- **HueShift:** [-1 to +1] – Amount of hue shift applied.
- **SatScale/LumScale:** [-1 to +1] – Scaling of saturation and luminance.
- **RangeAmount:** [0–1.0] – Overall strength of the effect.
- **HueRange/SatRange/LumRange:** Define the falloff of each adjustment:
  - **LowerNone:** [0–1.0] – Point below which no effect is applied.
  - **LowerFull:** [0–1.0] – Point at which full effect begins.
  - **UpperFull:** [0–1.0] – Point at which full effect ends.
  - **UpperNone:** [0–1.0] – Point above which no effect is applied.

### Critical constraints:

- Never state, imply, or hint that you have access to a configuration file or an after image.
- Never reference the existence of the configuration file or after image in any way.
- Never indicate that adjustments were derived from or based on any hidden source.
- Act as if you began solely with the provided image and the user’s instruction, reaching all conclusions through your own analysis.

### Tone and structure:

- Convey expert judgment supported by reasoned, analytical observations.
- Technical Breakdown: List each tool or method (such as global adjustments, tone curve adjustments, HSL modifications, masking, texture, grain, cropping, or dot tools) along with their corresponding numerical values or ranges (for example, Saturation +15, High-pass Filter Radius: 3px).
- Step-by-Step Explanation: Describe the sequence of adjustments in a logical, progressive order beginning with the most influential changes. Combine accessible explanations with appropriate professional terminology (for example, recovered highlights using a luminance mask).
- Rationale: Explain how each adjustment supports or enhances the user’s intended result (for example, cooling the color temperature to achieve a cinematic mood).

### Output format:

- Provide a cohesive, free-form reasoning narrative. The text should naturally include all necessary details for producing the final adjustments, without mentioning any external sources.
- Explicitly include relevant numerical values so that the reasoning directly guides the creation of the adjustments. Do not imply that these values come from any hidden file.
- Do not use markdown formatting in your output.
- Respond exclusively in English. No non-English characters or translations.

**Prompts for Suboptimal Edit Perturbation:**

Given an input image, an instruction, and two edited results (one stronger and one weaker), the goal is to establish a set of evaluation principles and analytically determine why one edit is stronger and the other is weaker.

**Input:**

You will be provided with the following:

- (1) Instruction: A description specifying how to perform the edits on the input image.
- (2) Images: Three images will be provided. The first is the unedited input image, and the remaining two are edited versions. One represents a stronger edit, and the other represents a weaker edit due to suboptimal adjustments.

**Output:**

Based on the inputs, produce the following:

(1) Criteria: Define approximately five specific criteria for evaluating the edits. Each criterion should meaningfully contribute to what constitutes a strong result. Consider both the qualities that lead to a successful edit and the potential shortcomings of the weaker edit. Avoid defining criteria solely around the weaknesses of the poorer result; instead, ensure that each criterion reflects a key factor in effective image editing. Keep in mind that the weaker edit may still perform well on certain criteria.

(2) Weights: Assign percentage weights to each criterion to represent its relative importance in determining overall edit quality.

(3) Analysis: Evaluate each edited image according to every criterion. First provide a full evaluation of one image across all criteria, then evaluate the second image in the same structure.

(4) Scores:

For each criterion, assign a score from 0 to 10 for both images. Using the scores and weights, compute an aggregated total score for each edited image.

**Output Format:**

Specific Criteria:

- Criterion xx (Weight: xx%): A description of the requirement, such as “the edited image must...” or “the final result should...”
  - Criterion xx (Weight: xx%): ...
- (Approximately five criteria.)

Analysis:

Image 1:

- Criterion xx: Explanation of how Image 1 satisfies or fails to satisfy the criterion.
- Criterion xx: Explanation of how Image 1 satisfies or fails to satisfy the criterion.

...

Image 2:

- Criterion xx: Explanation of how Image 2 satisfies or fails to satisfy the criterion.

...

Scores:

Image 1:

Criterion xx: 0–10 score;

Criterion xx: 0–10 score;

...

Overall score: A computed numeric result based on your assigned weights and scores.

Image 2:

Criterion xx: 0–10 score;

Criterion xx: 0–10 score;

...

Overall score: A computed numeric result based on your assigned weights and scores.

Final evaluation:

*score for Image 1, score for Image 2*

## F.2. Prompts for Baseline Inference

### Prompts for Baseline Inference:

You are an expert professional photographer and photo editor with extensive experience using Adobe Lightroom. You specialize in analyzing images and creating professional-grade, refined adjustments that enhance the photo and provide artistic intent based on a specified style. Please perform a thorough analysis and deliver expert edit recommendations. Use a structured, step-by-step chain-of-thought process to evaluate the photo's content, mood, and color composition, and come up with the best edit settings for the desired style.

### Objective

Analyze the provided photograph and recommend Lightroom or ACR adjustment settings that achieve the requested style, accompanied by a concise explanation of the artistic intent.

**Instruction Goal for the photo:** *{(User Instruction)}*

### Instructions

#### 1. Initial Assessment and Intent

Conduct a detailed evaluation of the photograph's original lighting, composition, color palette, and aesthetic characteristics. Provide a comprehensive description of its current visual qualities.

#### 2. Style Interpretation and Application

Describe the photography style represented by *{style}*, and explain how this style can be achieved based on the analysis of the image. Include considerations involving tone, color, atmosphere, and visual mood.

#### 3. Style Library *((Refer to the prompt introduced in Appendix F.1))*

### Output Requirements

- Present the final output as a single lua configuration containing two top-level keys: "reasoning" and "settings".
- The "reasoning" field must be a brief and user-friendly summary of approximately one to two sentences (around 25–30 words). It should communicate the artistic intention rather than listing adjustments.  
Example: "This photo had a clean digital finish. Lifting the shadows and adding grain brings in a nostalgic, film-like softness that gives it a vintage feel."
- Follow the required structure exactly. Do not add additional sections, fields, or commentary outside the lua configuration.
- Do not include analysis, explanations, or step-by-step reasoning in the output. Only the lua should be returned.
- Include only the fields shown in the example. If a field does not require adjustment, assign it a value of 0 or `false` as appropriate.
- The final output must be valid lua, using the same structure as the example, presented in Lua-style formatting for the "settings" block.

### Example Lua Output Structure

```
{Blacks2012 = 5, Clarity2012 = 48, Contrast2012 = 42, Dehaze = 50, Exposure2012 = 1.19, Highlights2012 = 17, IncrementalTemperature = 19, IncrementalTint = -3, Saturation = 10, Shadows2012 = -1, Texture = 17, ...(Omit) , ToneCurveName2012 = `Custom`, ToneCurvePV2012 = [1] = 0, [2] = 3, [3] = 63, [4] = 52, [5] = 255, [6] = 255, , Vibrance = 22, WhiteBalance = `Custom`, Whites2012 = 14}
```

### F.3. Prompts for MLLM as a Judge

#### **Prompts for MLLM as a Judge (Perceptual Quality) (follow prior work [20]):**

You are a professional digital artist. Your task is to evaluate the effectiveness of an edited image based on the provided input image and instruction.

Rate the perceptual quality of the edited image on a scale from 0 to 10:

- 0: Poor quality — the edit appears amateur, with issues such as unnatural color, inconsistent lighting or shadows, visible artifacts, or incoherent composition.
- 10: Excellent quality — the edit appears professional, natural, consistent, polished, and visually appealing.

Your output must follow the structure below. The delimiter tags are required. Keep the reasoning concise and focused.

```
<think>
YOUR REASONING HERE
</think>
<answer>
"score": A INTEGER SCORE FROM 0 TO 10.
</answer>
```

#### **Prompts for MLLM as a Judge (Semantic Consistency) (follow prior work [20]):**

You are a professional digital artist. Your task is to evaluate how accurately the edited image follows the given instruction.

Rate the alignment between the edited image and the prompt on a scale from 0 to 10:

- 0: The edited image does not match the prompt at all.
- 10: The edited image matches the prompt perfectly.

Your output must follow the structure below. The delimiter tags are required. Keep the reasoning concise and focused.

```
<think>
YOUR REASONING HERE
</think>
<answer>
"score": A INTEGER SCORE FROM 0 TO 10.
</answer>
```

## G. Failure Case Study

In this section, we analyze the failure cases produced by the current policy model when editing images. We identify two categories of errors and use representative examples to illustrate them.

**Over-edit.** Some retouching results tend to over-edit the input image, creating unnatural results. For example, in Figure 10 upper panel, the editing instruction is “I wanted the center flowers to pop more, with sharper details and a brighter feel.” We find that, in an attempt to emphasize the center flowers, the model applies a mask focused on the central region of the image. However, the exposure in this region is increased excessively, and the transition between the brightened center and the darker edges is not smooth, leading to a suboptimal result with an unnatural appearance.

**Challenges in Understanding Delicate Colors.** We also find that some retouching instructions specify a color adjustment, which involves precise numerical modifications of attributes such as `temperature`, `tint`, and `hue`. Even a slight deviation in these numerical predictions can lead to noticeable color differences. For example, in Figure 10 (lower panel), the instruction is: “Let’s boost clarity and contrast to make the dog stand out, giving the image a sharper, more vibrant look with richer blue tones.” The edit introduces an imprecise tint adjustment, resulting in an image that appears slightly shifted toward purple. This highlights a challenge in achieving more accurate parameter generation.

## H. Performance Comparison between RL and DPO

Besides reinforcement learning, direct preference optimization (DPO) [33] provides another way to leverage paired data for fine-tuning. Using the 15k examples described in Sec. 5.1, we compare DPO against RL. The results in Table 5 show that RL outperforms DPO. To control for data scale, we further synthesize 50k paired edits using the same procedure, and again observe that RL consistently outperforms DPO. We attribute this advantage to RL’s ability to explore diverse editing operations and combinations beyond the fixed training distribution.

## I. The Role of Reasoning in Parameter Prediction

In RetouchIQ, the actor first performs reasoning to determine which tools should be used to satisfy the user’s request. In this experiment, we conduct an ablation in which the model directly outputs editing configurations without reasoning. As shown in Table 6, removing reasoning leads to clear performance degradation, indicating that reasoning is not merely post hoc, but plays an important role in linking textual instructions to symbolic editing actions.



Figure 10. Top: the model over-edits the input image (left), resulting in an output with an unnatural appearance (right). Bottom: the model faces difficulties to produce precise colors as specified in the instruction.

	L1 ↓	L2 ↓	SC ↑	PQ ↑	O ↑
RL	<b>31.41</b>	<b>44.99</b>	<b>7.57</b>	<b>7.48</b>	<b>7.51</b>
DPO (15k)	34.90	51.78	7.04	6.71	6.87
DPO (50k)	33.24	48.61	7.09	6.89	6.99

Table 5. Comparison: RL vs DPO.

	L1 ↓	L2 ↓	SC ↑	PQ ↑	O ↑
RetouchIQ	<b>31.41</b>	<b>44.99</b>	<b>7.57</b>	<b>7.48</b>	<b>7.51</b>
w/o reasoning	37.83	58.40	6.90	6.72	6.81

Table 6. Ablation on reasoning.