EDITTHINKER: ENHANCING IMAGE EDITORS VIA ITERATIVE REASONING WITH A PLUG-AND-PLAY VLM

Anonymous authors
Paper under double-blind review

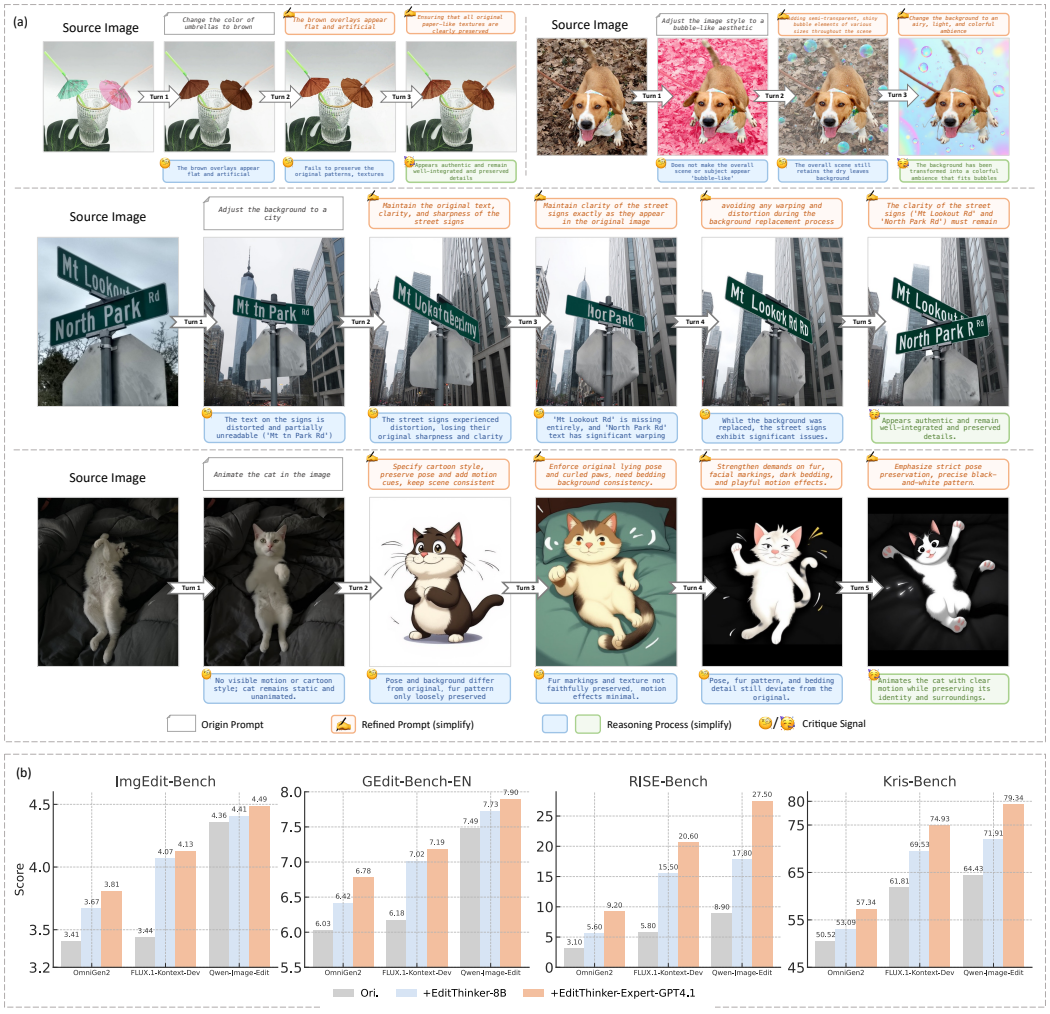

Figure 1: **Overview of EditThinker.** Subfigure (a) illustrates how a trained VLM critiques editing results, reasons about failures, and refines instructions through a **Critique-Refine-Repeat** loop with a fixed image editor. Subfigure (b) reports the resulting improvements on image editing benchmarks.

ABSTRACT

Image editors can produce realistic images while still failing to satisfy the requested changes. Improving their instruction following requires identifying these failures and translating visual feedback into instructions that the editor can execute. We introduce **EditThinker**, an 8B vision-language model (VLM) that enhances existing image editors through iterative reasoning while keeping their

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

weights fixed. EditThinker jointly critiques semantic consistency and visual quality, reasons about editing failures, and refines the instruction in a structured response. It interacts with the editor through a *Critique-Refine-Repeat* loop, providing a plug-and-play interface for feedback-driven editing. Learning effective refinements requires more than imitating plausible expert suggestions: the instructions must work with the capabilities of the underlying editor. We therefore combine supervised fine-tuning on expert demonstrations with reinforcement learning from executed editing outcomes, using format, critic, and edit rewards. To support this training, we construct THINKEDIT-167k, a dataset of 167k step-wise samples with unified supervision for critique, reasoning, and instruction refinement. Experiments on four benchmarks show improved overall instruction-following scores across the tested editors. Ablations examine the joint model design, data allocation, and training across VLM backbones. Our round-budget study further shows a favorable quality-editor-call trade-off relative to best-of-N selection. We will release our data construction framework, datasets, and models.

1 INTRODUCTION

Instruction-based image editing modifies a source image according to natural language instructions while preserving content that should remain unchanged. Recent editors built on strong generative models (Wu et al., 2025a; Batifol et al., 2025; Wu et al., 2025c; Team et al., 2025; AI, 2024) produce realistic images, yet can still misinterpret requested changes or alter unrelated regions. Addressing these failures requires understanding both the intended modification and the actual editing result. A reasoning module must translate visual feedback into instructions the editor can execute.

Existing strategies improve the inputs or select among outputs of an editor. Prompt enhancement (Wang et al., 2025b) improves instructions, while best-of-N selection with a scoring model such as EditScore (Luo et al., 2025) searches among candidate edits. Iterative feedback is established in language-model refinement (Madaan et al., 2023; Shinn et al., 2023) and visual generation and editing. Idea2Img (Yang et al., 2023) and GenArtist (Wang et al., 2024) use proprietary vision-language models (VLMs) for planning and feedback, while ReasonEdit (Yin et al., 2025) trains both a reasoner and an editing model. We focus on learning an editing-oriented VLM that turns failures into actionable revisions while keeping the editor fixed. This setting allows the reasoning module to guide existing editors through their instruction interface, without adapting their generative weights. Appendix A.2 discusses related refinement approaches.

The central challenge is to make reasoning useful for execution. A VLM may correctly describe an editing error yet suggest a revision that the same editor cannot realize. Because the editor is fixed, the thinker must learn which changes to the instruction lead to better visual outcomes. We therefore connect two learning objectives: understanding the current edit and producing a refinement that works when executed. This motivates a joint Critic-Refiner design and a training procedure that combines expert demonstrations with feedback from actual edits.

We introduce **EditThinker**, an 8B VLM trained to jointly critique editing results and refine instructions. Given the source image, original instruction, previous instruction, and current result, it generates a reasoning trace, semantic-consistency and perceptual-quality scores, and a revised instruction. Critique and refinement share a single structured response, with reasoning generated before the revised instruction. This design connects assessment of the visual result to the next editing action. We use a **Critique-Refine-Repeat** loop to execute these actions: the fixed editor applies the revised instruction to the source image, and the thinker inspects the new result. The loop stops at a score threshold or iteration budget.

Training this module requires both demonstrations of useful reasoning and feedback about whether its instructions work in practice. We collect expert-guided instruction-refinement trajectories with existing editors, pairing editing states with reasoning traces and revised instructions. An additional scorer evaluates intermediate results; filtering and truncation select trajectories that match or improve on the initial edit, from which we extract **THINKEDIT-167k** step-wise samples. We partition the data by training objective, using stable or consistently high-quality samples for supervised fine-tuning (SFT) and trajectories with high score variance for reinforcement learning (RL). The resulting 140k SFT and 27k RL samples connect source images, editing states, reasoning traces, and refined

instructions. This division separates learning stable refinement patterns from optimizing decisions on trajectories with varying outcomes. Random and reversed data allocations provide alternative configurations for examining this choice across the two training stages.

SFT initializes the thinker with the expert's reasoning and structured output format. RL then updates the thinker from actual editor executions, using rewards for format adherence, critique, and editing improvement. This second stage addresses the gap between imitating an expert's instruction and producing one that works with a fixed editor. We study the resulting quality-iteration trade-off: different reward configurations can favor editing quality or fewer editor calls, and combining rewards offers a balance in our experiments. The editor remains fixed throughout this training and refinement process. At inference, the trained 8B thinker provides both critique and instruction refinement, interacting with existing editors through their image-and-text interface.

We evaluate EditThinker on four benchmarks covering general and reasoning-based image editing. It improves aggregate scores across the tested editors. Ablations examine rewards, iteration budgets, role coupling, reasoning order, and the SFT/RL data partition. Paired comparisons of additional VLM backbones before and after EditThinker training assess the applicability of the training procedure. Round-budget comparisons also show fewer editor calls than best-of-N selection.

Our contributions are:

1. An 8B plug-and-play VLM that jointly critiques and refines editing instructions through explicit reasoning while preserving the underlying editor weights.
2. THINKEDIT-167k, a dataset of step-wise refinement samples, and an SFT/RL training approach combining expert demonstrations with feedback from executed edits.
3. Evaluation across multiple editors, with design and data-partition ablations and paired backbone-training comparisons, examining editing quality and editor-call trade-offs.

2 EDITTHINKER

EditThinker is an 8B VLM trained to critique editing results and refine instructions in a *Critique-Refine-Repeat* loop while keeping the underlying image editor fixed.

2.1 OVERALL FRAMEWORK

A single-pass editor takes a source image I_{src} and the original instruction T_s to produce an edited image. EditThinker adds feedback to this process by inspecting the result and refining the instruction for a subsequent editing round.

The workflow separates a VLM-based Thinker for critique and instruction refinement from an Editor for execution (e.g., Qwen-Image-Edit or FLUX.1 Kontext). We train the Thinker via SFT and RL, and use the Editor through its image-and-text interface without updating its parameters. At iteration t, the Thinker inspects the previous output I_{edit}^{t-1} and generates reasoning R_t, semantic and quality scores S_t, and a refined instruction T_t in one structured response:

$$(S_t, R_t, T_t) = \text{Thinker}(I_{src}, I_{edit}^{t-1}, T_{t-1}, T_s). \tag{1}$$

Then the Editor executes the new instruction T_t on the source image I_{src}, generating the updated result I_{edit}^{t} as:

$$I_{edit}^{t} = \text{Editor}(I_{src}, T_t). \tag{2}$$

We stop when the geometric mean of the predicted semantic and quality scores reaches $S_{\text{thr}} = 8$, or after $N = 5$ editing rounds by default.

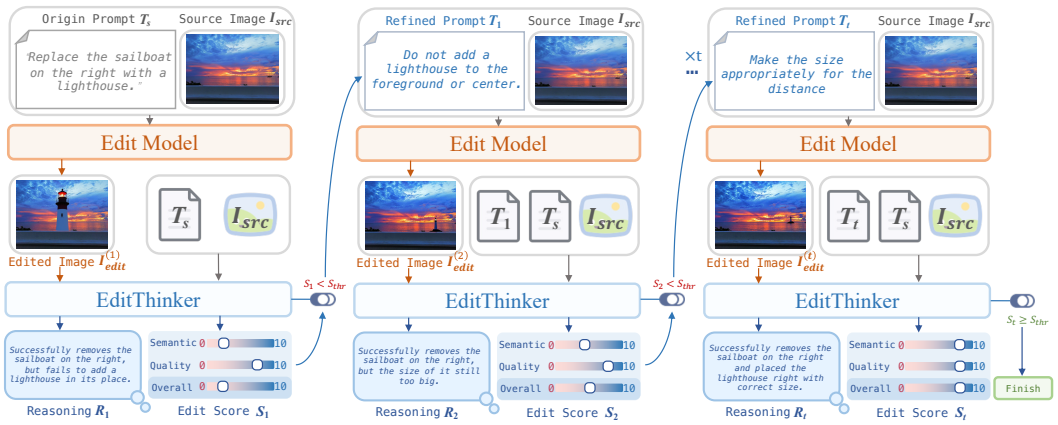

Figure 2: **Critique-Refine-Repeat pipeline.** The editor first applies the original instruction T_s to the source image I_{src}. Given this initial edit, EditThinker generates score S_t, reasoning R_t, and refined instruction T_t. The fixed editor applies T_t to I_{src} until the stopping threshold or iteration budget is reached.

2.2 Design of the EditThinker

We formulate EditThinker as a **dual-role model** that jointly evaluates and plans. Its Critic and Refiner share the same VLM and produce one structured response, allowing instruction refinement to use the critique of the current editing result.

We order the output so that the model reasons about the previous edit and generates its scores before producing the refined instruction T_t. Generating R_t before T_t lets the instruction condition on the visual critique of I_{src} and I_{edit}^{t-1}. We evaluate both role coupling and output order in table 3c.

To implement this dual-role design, we define a structured input-output format that explicitly encodes the evaluation-then-planning process.

Input Tuple. EditThinker receives a multimodal tuple $(I_{src}, I_{edit}^{t-1}, T_s, T_{t-1})$ at each iteration t, providing complete context of the editing state: I_{src} and T_s represents the original reference, I_{edit}^{t-1} is the current result to be critiqued, and T_{t-1} is the previous instruction that produced it.

Structured Output Format. The output is a structured text string that serializes EditThinker's reasoning process:

<think> Reasoning process... </think>
<score> [S_{sem}, S_{qual}] </score>
<answer> Refined prompt T_t </answer>

Here, S_{qual} is the perceptual quality of I_{edit}^{t-1}, and S_{sem} is the semantic alignment with the original instruction T_s relative to I_{src}.

2.3 Training of EditThinker

Training EditThinker to perform this dual-role task requires a specialized dataset and a multi-stage training strategy. We adopt a two-stage approach: first, supervised fine-tuning (SFT) to learn the output format and basic reasoning, followed by reinforcement learning (RL) to optimize instruction refinement based on actual editing feedback. The data construction process is detailed in section 3.

2.3.1 SUPERVISED FINE-TUNING (COLD START)

Using the expert (GPT-4.1) demonstration dataset (detailed in section 3), the base VLM learns to adopt our structured I/O format (e.g., <think>, <score>, <answer>), mimic the expert's reasoning style, and understand the principles of critiquing and refining instructions.

2.3.2 REINFORCEMENT LEARNING TUNING (RLT)

SFT imitates expert reasoning but may produce instructions that fail under the practical limitations of specific editors. We therefore optimize EditThinker with GRPO (Group Relative Policy Optimization) using actual editing feedback and a multi-component reward for its critic and refiner roles (section 2.2).

Critic Reward. We use GPT-4.1 (OpenAI, 2025a) as the critic expert. For each dimension $d \in \{\text{sem}, \text{qual}\}$, we compare the Thinker's predicted score $S_{d,t}$ with the expert score $\mathcal{E}_d$ of the same current edit I_{edit}^{t-1}, taking the negative mean absolute error:

$$R_{\text{critic}} = -\frac{1}{2} \sum_{d \in \{\text{sem},\text{qual}\}} \left| S_{d,t} - \mathcal{E}_d(I_{src}, I_{edit}^{t-1}, T_s) \right|. \tag{3}$$

Both the Thinker and the expert assess I_{edit}^{t-1}, before executing the new instruction T_t. This reward trains the critique of the current result; the edit reward below evaluates the subsequent editing improvement.

Edit Reward. To train instruction refinement, we reward improvement between successive edits evaluated by the same expert:

$$R_{\text{edit}} = \mathcal{E}(I_{src}, I_{edit}^{t}, T_s) - \mathcal{E}(I_{src}, I_{edit}^{t-1}, T_s) \tag{4}$$

The reward is positive only when T_t produces a better edit, grounding refinement in execution feedback.

The final reward R_{total} is as follows:

$$R_{overall} = \alpha R_{format} + \beta R_{critic} + \gamma R_{edit} \tag{5}$$

where R_{format} is the reasoning format reward and $(\alpha, \beta, \gamma) = (0.5, 0.1, 0.4)$. Design ablations appear in table 3c; reward-weight ablations are in Appendix C.

3 THINKEDIT DATASET

To train the EditThinker, we require a high-quality dataset that captures the multi-turn *Critique-Refine-Repeat* cycle. As shown in fig. 3, we designed an automated data construction pipeline to simulate this process, consisting of four sequential steps: Trajectory Generation, Trajectory Filter, Step-wise Filter, and Data Partition. This pipeline allowed us to construct our **THINKEDIT-167k** dataset. We detail each step below.

3.1 TRAJECTORY GENERATION

The first stage focuses on simulating the multi-turn "Critique-Refine-Repeat" cycle. The pipeline begins with an Edit Data Pool containing diverse (I_{src}, T_s) pairs. At each step t, the edit thinker expert (GPT-4.1) evaluates the current state (based on I_{src}, T_s, and I_{edit}^{t-1}) and generates a new instruction (T_t), reasoning process (R_t) and ⟨stop⟩ token.

Notably, the expert does not output a score (S_t). Instead, it directly determines when to halt the process by issuing a ⟨stop⟩ token. This design choice stems from our finding that a single expert struggles to maintain high performance in both task refinement and output scoring simultaneously.

If a ⟨stop⟩ token is not issued, the image editor uses the new T_t to produce I_{edit}^{t}. This loop continues until the expert triggers the ⟨stop⟩ condition (or a max-iteration limit N is hit), thus completing a full trajectory.

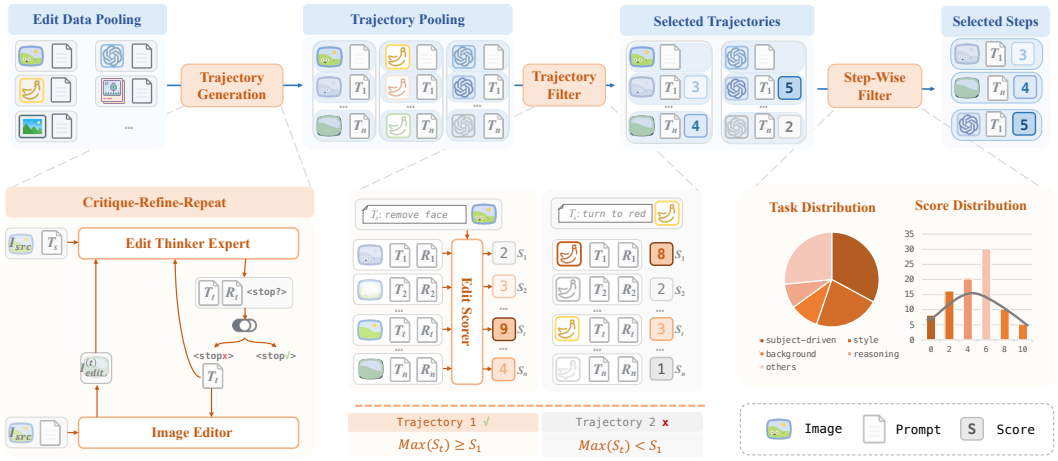

Figure 3: **ThinkEdit data construction.** Expert-guided editing generates trajectories, which are scored, filtered for improvement, and truncated at their best step. We extract step-wise samples, balance task and score distributions, and partition them for SFT and RL.

3.2 Trajectory Filter

Since the edit thinker expert only generates refined instructions and a ⟨stop⟩ token without quality scores, we employ an additional edit scorer to evaluate each step $I_{edit}^{(t)}$ and assign a score S_t. After scoring all steps $(S_1, \ldots, S_n)$, we apply a two-stage filtering process:

Filter Failed Trajectories. We retain only trajectories where at least one subsequent step ($t > 1$) achieves a score higher than or equal to the initial step (i.e., $\max(S_{t>1}) \geq S_1$). Trajectories failing this condition are discarded.

Truncate Kept Trajectories. For retained trajectories, we identify the step k with the highest score ($S_k = \max(S_{t\geq 1})$) and truncate the trajectory to include only steps from 1 to k. All subsequent steps ($t > k$) are discarded.

3.3 Step-wise Filter

Finally, we process the curated trajectories from the Trajectory Filter to create the final training data through two steps:

Sample Extraction. First, we unroll the truncated trajectories. Each individual step t within a trajectory is converted into a distinct training sample. This sample pairs an input tuple (I_{src}, I_{edit}^{t-1}, T_s, T_{t-1}) with its corresponding ground-truth expert output (R_t, T_t), retaining the score S_t as metadata for subsequent filtering.

Distribution Balancing. We filter samples to balance task types (e.g., object removal, color modification, and addition) and score levels, ensuring coverage of both editing tasks and output quality.

3.4 SFT and RL Data Split

We partition the curated data by training objective: stable, high-quality examples for SFT and challenging improvement trajectories for RL.

RL Dataset. We select 10k trajectories with high score variance ($\text{Var}(S_t) > \theta$, $\theta = 3.0$), balancing task and score distributions. These challenging trajectories provide informative improvement signals and yield 27k step-wise RL samples.

SFT Dataset. We select 140k step-wise samples with low score variance or consistently high quality, providing reliable demonstrations of stable instruction refinement.

Table 1: Results on ImgEdit-Bench and GEdit-Bench-EN. ‡ marks our single-pass evaluation of official editor checkpoints without editor fine-tuning; paired gains use these reproduced baselines.

Model	ImgEdit-Bench										GEdit-Bench-EN		
	Add	Adjust	Extract	Replace	Remove	Background	Style	Hybrid	Action	Overall	G_SC	G_PQ	G_O
Open-source Models													
IP2P (Brooks et al., 2023)	2.45	1.83	1.44	2.01	1.50	1.44	3.55	1.20	1.46	1.88	3.58	5.49	3.68
AnyEdit (Jiang et al., 2025)	3.18	2.95	1.88	2.47	2.23	2.23	2.85	1.56	2.65	2.45	3.18	5.82	3.21
UltraEdit (Zhao et al., 2024)	3.44	2.81	2.13	2.96	1.45	2.86	3.76	1.91	2.98	2.70	-	-	-
OmniGen (Xiao et al., 2025)	3.47	3.04	1.71	2.94	2.43	3.21	4.19	2.24	3.38	2.96	5.96	5.89	5.06
Step1X-Edit (Liu et al., 2025a)	3.88	3.14	1.76	3.40	2.41	3.16	4.63	2.64	2.52	3.06	7.66	7.35	6.97
ICEdit (Zhang et al., 2025b)	3.58	3.39	1.73	3.15	2.93	3.08	3.84	2.04	3.68	3.05	-	-	-
BAGEL (Deng et al., 2025)	3.56	3.31	1.70	3.30	2.62	3.24	4.49	2.38	4.17	3.20	7.36	6.83	6.52
OmniGen2 (Wu et al., 2025c)	3.57	3.06	1.77	3.74	3.20	3.57	4.81	2.52	4.68	3.44	7.16	6.77	6.41
Ovis-U1 (Wang et al., 2025a)	4.13	3.62	2.98	4.45	4.06	4.22	4.69	3.45	4.61	4.00	-	-	6.42
FluxKontext dev (Labs et al., 2025b)	3.76	3.45	2.15	3.98	2.94	3.78	4.38	2.96	4.26	3.52	6.52	7.38	6.00
UniWorld-V2 (Li et al., 2025b)	4.29	4.44	4.32	4.69	4.72	4.41	4.91	3.83	4.83	4.49	8.39	8.02	7.83
Proprietary Models													
GPT-4o (OpenAI, 2024)	4.61	4.33	2.90	4.35	3.66	4.57	4.93	3.96	4.89	4.20	-	-	7.49
Critique-Refine-Repeat													
OmniGen2‡	3.91	3.23	2.03	2.84	3.11	3.94	4.59	2.76	4.69	3.41	6.47	7.04	6.03
+ EditThinker-8B	4.50	3.30	3.33	3.60	2.55	4.16	4.31	3.42	4.76	**3.67**	6.65	7.44	**6.42**
+ EditThinker-Expert-GPT4.1	4.21	3.28	3.04	3.80	3.39	4.16	4.61	2.97	3.39	3.81	7.34	7.24	6.78
Flux-Kontext-dev‡	3.83	3.55	2.18	3.91	2.74	3.79	4.42	2.82	4.18	3.44	6.62	7.61	6.18
+ EditThinker-8B	4.20	4.03	3.21	4.55	3.65	3.52	4.56	4.44	4.54	**4.07**	7.59	7.63	**7.02**
+ EditThinker-Expert-GPT4.1	4.08	4.01	3.45	4.44	3.75	4.19	4.59	3.73	4.57	4.13	7.83	7.66	7.19
Qwen-Image-Edit‡	4.59	4.32	3.79	4.57	3.86	4.54	4.83	3.85	4.70	4.36	8.01	7.87	7.49
+ EditThinker-8B	4.47	4.39	3.80	4.61	4.42	4.33	4.69	3.00	4.85	**4.41**	8.30	7.86	**7.73**
+ EditThinker-Expert-GPT4.1	4.47	4.27	4.18	4.58	4.59	4.55	4.81	3.72	4.77	4.49	8.57	7.86	7.90
Longcat-Image-Edit‡	4.55	4.33	4.26	4.51	4.47	4.43	4.73	4.62	4.91	4.53	8.25	8.17	7.81
+ EditThinker-8B	4.58	4.46	4.38	4.15	4.85	4.64	4.76	3.25	4.89	**4.57**	8.44	8.17	**8.00**
+ EditThinker-Expert-GPT4.1	4.60	4.68	3.75	4.52	4.69	4.36	4.71	4.67	4.53	4.54	8.63	8.21	8.12

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation Detail. EditThinker uses Qwen3-VL-8B-Instruct (Bai et al., 2025). We train on THINKEDIT-SFT-140k for one epoch with learning rate 2×10^{-5} and batch size 32, then on THINKEDIT-RL-27k for one epoch with learning rate 2×10^{-6}, global batch size 16, GRPO rollout group size 8, and KL coefficient 10^{-3}. MAX_PIXELS is 1024×1024. Training uses 8 H800 GPUs for approximately 48 hours. At inference, we keep OmniGen2 (Wu et al., 2025c), FLUX.1-Kontext [Dev] (Batifol et al., 2025), Qwen-Image-Edit (Wu et al., 2025a), and Longcat-Image-Edit (Team et al., 2025) fixed. We use at most $N = 5$ editing rounds and stop early when the geometric mean of the semantic and quality scores reaches 8. Appendix D summarizes the hyperparameters.

Benchmarks and Baselines. To evaluate how EditThinker improves existing editors, we use four benchmarks: ImgEdit-Bench (Ye et al., 2025), GEdit-Bench (Liu et al., 2025b), RISEBench (Zhao et al., 2025), and KRIS-Bench (Wu et al., 2025e). Together, they cover both general editing and edits requiring reasoning, with RISEBench and KRIS-Bench specifically testing the latter. We compare each fixed editor with its single-pass baseline and the GPT-4.1 expert-thinker variant.

4.2 MAIN RESULTS

We evaluate our EditThinker framework across a comprehensive suite of four benchmarks to assess its performance on both general and reasoning-based editing tasks. For general image editing, we use ImgEdit-Bench and GEdit-Bench-EN (results in table 1). For reasoning-based editing, we report RISE-Bench results in table 2; KRIS-Bench results and analysis are provided in Appendix B.

Performance on General Editing. As shown in table 1, EditThinker improves the overall ImgEdit-Bench and GEdit-Bench-EN scores of the four tested editors. These aggregate gains do not imply improvement on every editing category; failure cases and limitations are discussed in Appendix G. The paired comparisons use the single-pass baselines marked ‡, which were evaluated using the corresponding official editor checkpoints without fine-tuning.

Performance on Reasoning Editing. Crucially, our method's advantages are not limited to general edits; it provides equally consistent improvements on tasks requiring deep reasoning. As detailed in table 2, on RISE-Bench, which tests complex spatial, causal, and temporal reasoning, our framework provides a stable performance lift for all models and shows significant improvements in their reasoning capabilities.

Comparison with the Expert Thinker. We compare the trained 8B VLM with GPT-4.1 as alternative thinkers for the same fixed editor. This comparison examines the editing quality achieved by EditThinker relative to the expert used to generate its demonstrations. The tables report both variants for each editor, including FLUX.1-Kontext [Dev]. On ImgEdit-Bench, EditThinker-8B improves the FLUX score to 4.07, while the stronger EditThinker (GPT-4.1) boosts it even further to 4.13. The expert improves the FLUX results here, although a stronger expert does not uniformly win on every editor and benchmark. We evaluate additional VLM backbones in table 3d and a non-GPT evaluator in table 4.

Table 2: Comparison of model performance on RISE-Bench. ‡ indicates results from our own tests with official model.

Model	Temporal	Causal	Spatial	Logical	Overall
Proprietary Models					
Seedream-4.0 (ByteDance, 2025b)	12.9	12.2	11.0	7.1	10.8
GPT-Image-1 (OpenAI, 2025)	34.1	32.2	37.0	10.6	28.9
Gemini-2.5-Flash-Image (Google, 2025)	25.9	47.8	37.0	18.8	32.8
Open-source Models					
Step1X-Edit (Liu et al., 2025b)	0.0	2.2	2.0	3.5	1.9
Ovis-U1 (Wang et al., 2025a)	1.2	3.3	4.0	2.4	2.8
FLUX.1-Kontext-Dev (Batifol et al., 2025)	2.3	5.5	13.0	1.2	5.8
BAGEL (Deng et al., 2025)	2.4	5.6	14.0	1.2	6.1
Qwen-Image-Edit (Wu et al., 2025a)	4.7	10.0	17.0	2.4	8.9
BAGEL (w/ CoT) (Deng et al., 2025)	5.9	17.8	21.0	1.2	11.9
Critique-Refine-Repeat					
OmniGen2‡	2.4	1.1	7.0	1.2	3.1
+ EditThinker-8B	3.5	10.0	5.0	3.5	**5.6**
+ EditThinker-Expert-GPT4.1	17.6	8.9	8.0	2.4	9.2
FLUX.1-Kontext-Dev‡	2.3	5.5	13.0	1.2	5.8
+ EditThinker-8B	12.9	16.6	25.0	5.8	**15.5**
+ EditThinker-Expert-GPT4.1	16.5	24.4	33.0	5.9	20.6
Qwen-Image-Edit‡	4.7	10.0	17.0	2.4	8.9
+ EditThinker-8B	10.8	23.3	27.0	8.2	**17.8**
+ EditThinker-Expert-GPT4.1	25.9	32.2	40.0	9.4	27.5
Longcat-Image-Edit‡	9.4	15.6	17.0	4.7	11.9
+ EditThinker-8B	27.1	28.9	24.0	8.2	**22.2**
+ EditThinker-Expert-GPT4.1	27.1	40.0	23.0	11.7	25.5

4.3 Ablation Study

We use FLUX.1-Kontext [Dev] as the editor to study training, inference, thinker design, and backbone choices.

Training Strategy. We test how SFT and the two RL rewards affect quality and editing rounds in table 3a. SFT improves GEdit from 6.43 to 6.92. Critic-only RL reduces rounds to 2.13 but lowers the score to 6.74; edit-only RL reaches 6.95 at 3.93 rounds. Combining both rewards reaches 7.02 at 2.19 rounds, improving both measures over SFT alone (6.92 at 2.55). The trade-off depends on the benchmark: edit-only RL scores higher on ImgEdit (4.15 versus 4.07), but requires more calls (3.70 versus 1.85). Direct RL without SFT gives 6.51 on GEdit, supporting the SFT cold start.

Editing-Round Budget. Table 3b compares iterative refinement with EditScore (Luo et al., 2025) best-of-N sampling. EditThinker-8B gains 0.12 GEdit points from budgets 2 to 4, but only 0.02 from 8 to 10, indicating uneven marginal returns from additional budget. At budget 10, it scores 7.16 with 2.44 average calls, versus 7.31 with 3.38 for the GPT-4.1 expert and 6.67 with 10 for EditScore. Refinement thus outperforms selection while using fewer editor calls; these counts exclude VLM computation and do not measure total latency or FLOPs.

Thinker Design and Data Partition. We examine role coupling and output order separately in table 3c. Decoupling the Critic and Rewriter reduces GEdit from 7.02 to 6.90 and increases average rounds from 2.19 to 2.45. Generating the instruction before reasoning (T-before-R) instead yields 6.84. Both variants also lower ImgEdit scores (3.88 and 3.82 versus 4.07). These comparisons favor the evaluated joint, reasoning-first design, with the decoupled variant requiring more editor calls despite lower quality.

The split ablations examine the assignment of stable or high-quality examples to SFT and high-variance trajectories to RL (section 3). Random splitting improves during RL, from 6.92/3.98 to 6.97/4.01 on GEdit/ImgEdit. Reversing the allocation instead lowers scores from 6.95/3.95 to 6.87/3.84, even as average rounds decrease. The GEdit ranking of random and reversed splits therefore reverses after RL: the stronger SFT result does not imply a stronger final model. Both final variants trail our 7.02/4.07, favoring the proposed partition over both alternatives after the complete training sequence.

Generalization Across Training Backbones. Table 3d tests whether the training gains extend beyond the default backbone. Each trained variant is compared with its corresponding backbone without EditThinker training. All three improve across all benchmarks; QwenVL2.5-7B rises from 6.26/3.44/10.0 to 6.93/4.06/12.7 on GEdit/ImgEdit/RISE. The consistent within-backbone gains support applying the training procedure to different VLMs, although improvement magnitudes vary.

Table 3: Ablations and backbone comparisons with FLUX.1-Kontext [Dev]. Rounds denotes average editor calls. In (c), random/reversed SFT + RL follows the corresponding SFT stage. Scores are higher-is-better.

(a) Ablation Study on GEdit-Bench-EN and ImgEdit-Bench with SFT and RL Rewards

SFT	RL		GEdit-Bench-EN				ImgEdit-Bench	
	R_{critic}	R_{edit}	G_SC↑	G_PQ↑	G_O	Rounds↓	Overall↑	Rounds↓
FLUX.1-Kontext [Dev]			6.62	7.61	6.18	1.00	3.36	1.00
✗	✗	✗	6.91	7.64	6.43	1.99	3.64	1.91
✓	✗	✗	7.53	7.58	6.92	2.55	3.98	2.24
✓	✓	✗	7.34	7.50	6.74	2.13	4.01	1.94
✓	✗	✓	7.43	7.65	6.95	3.93	4.15	3.70
✗	✓	✓	6.64	7.95	6.51	2.00	3.75	1.99
✓	✓	✓	7.59	7.63	7.02	2.19	4.07	1.85

(b) Ablation Study on Max Turn Number and Performance Efficiency

Max Turn	EditThinker-8B		Expert-GPT 4.1		EditScore-8B	
	G_O↑	Rounds↓	G_O↑	Rounds↓	G_O↑	Rounds↓
1	6.18	1.00	6.18	1.00	6.18	1.00
2	6.93	1.67	6.95	1.46	6.41	2.00
4	7.05	2.07	7.13	2.05	6.59	4.00
6	7.06	2.24	7.16	2.53	6.62	6.00
8	7.14	2.35	7.27	2.97	6.64	8.00
10	7.16	2.44	7.31	3.38	6.67	10.00

(c) Ablations of the Thinker Design and Data Partition on GEdit-Bench-EN and ImgEdit-Bench

Configuration	GEdit-EN				ImgEdit	
	G_{SC} ↑	G_{PQ} ↑	G_O ↑	Rounds↓	Overall↑	Rounds↓
Ours	7.59	7.63	7.02	2.19	4.07	1.85
Decoupled roles	7.42	7.61	6.90	2.45	3.88	2.03
T-before-R	7.36	7.55	6.84	2.25	3.82	1.95
Random: SFT	7.48	7.59	6.92	2.60	3.98	2.26
Random: SFT + RL	7.52	7.60	6.97	2.24	4.01	2.03
Reversed: SFT	7.49	7.60	6.95	2.50	3.95	2.19
Reversed: SFT + RL	7.38	7.56	6.87	2.36	3.84	2.08

(d) Generalization across VLM backbones. ET denotes EditThinker training (✓: with; ✗: without).

Backbone	ET	GEdit-EN	ImgEdit	RISE
FLUX.1-Kontext [Dev]		6.18	3.36	5.8
InternVL2.5-8B	✗	6.22	3.38	8.1
	✓	6.84	3.54	11.1
InternVL3-8B	✗	6.32	3.40	9.4
	✓	6.88	3.68	12.5
QwenVL2.5-7B	✗	6.26	3.44	10.0
	✓	6.93	4.06	12.7

Evaluation with Gemini-2.5-Pro. We assess whether the training improvements also appear under a non-GPT judge in table 4. Gemini-2.5-Pro scores increase from 5.08/3.07 for the base FLUX editor to 5.38/3.23 with Qwen3VL-8B without EditThinker training, 5.59/3.36 after SFT, and 5.74/3.44 after SFT and RL on GEdit/ImgEdit. The GPT-4.1 expert reaches 5.90/3.46 under this evaluator. Thus, the ordering across these training stages is preserved with a different evaluator, lending support to the measured improvements beyond the primary judge. Absolute score levels remain specific to the evaluator.

Table 4: Gemini-2.5-Pro evaluation with FLUX.1-Kontext. Higher scores are better.

Configuration	GEdit-Bench-EN	ImgEdit-Bench
FLUX.1-Kontext	5.08	3.07
+ Qwen3VL-8B	5.38	3.23
+ SFT	5.59	3.36
+ SFT + RL	5.74	3.44
+ Expert-GPT4.1	5.90	3.46

Appendix C provides the ReasonEdit reference comparison, reward-weight sensitivity, prompt-enhancement and expert-model comparisons, and expert joint-scoring analysis; Appendix G presents failure and difficult cases.

5 Conclusion

We present EditThinker, a plug-and-play 8B VLM that enhances fixed image editors through iterative reasoning. SFT and RL train the model to jointly critique results and refine instructions using expert demonstrations and executed editing feedback. Experiments show improved overall instruction-following scores across four benchmarks and the tested editors, with fewer editor calls than best-of-N selection in our budget study. Ablations support joint critique and refinement, reasoning before instruction generation, and the proposed allocation of data between SFT and RL. Consistent improvements across separately trained VLM backbones further suggest that the training approach extends beyond a single backbone. The gains are not uniform across subtasks, and some edits remain unresolved even with additional iterations. The additional ablations and failure cases clarify both the benefits and the limits of this approach.

AI USE STATEMENT

We used generative AI models for data construction and evaluation, as described in the paper. We also used AI tools to assist with research discussions, result interpretation, figure preparation, and manuscript writing and translation. All content was reviewed and verified by the authors, who take responsibility for the final work.

REPRODUCIBILITY STATEMENT

Code and data will be released upon acceptance. The main paper and appendix describe the data construction procedure, training objectives, hyperparameters, stopping criteria, and evaluation protocols. These details, together with the released implementation and trajectories, will facilitate independent verification and reproduction of our results.

REFERENCES

Stability AI. Sd3-medium. https://stability.ai/news/stable-diffusion-3-medium, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, 2025.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023.

ByteDance. Doubao: Bytedance's ai chat assistant. https://www.doubao.com/chat/, 2025a. Accessed: 2025-05-08.

ByteDance. Seedream 4.0. 2025b. URL https://seed.bytedance.com/en/seedream4_0/.

Qi Cai, Jingwen Chen, Yang Chen, Yehao Li, Fuchen Long, Yingwei Pan, Zhaofan Qiu, Yiheng Zhang, Fengbin Gao, Peihan Xu, et al. Hidream-i1: A high-efficient image generative foundation model with sparse diffusion transformer. *arXiv preprint arXiv:2505.22705*, 2025.

Di Chang, Mingdeng Cao, Yichun Shi, Bo Liu, Shengqu Cai, Shijie Zhou, Weilin Huang, Gordon Wetzstein, Mohammad Soleymani, and Peng Wang. Bytemorph: Benchmarking instruction-guided image editing with non-rigid motions. *arXiv preprint arXiv:2506.03107*, 2025.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*, 2024.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Yuan Gong, Xionghui Wang, Jie Wu, Shiyin Wang, Yitong Wang, and Xinglong Wu. Onereward: Unified mask-guided image generation via multi-task human preference learning. *arXiv preprint arXiv:2508.21066*, 2025.

Google. Nano banana. 2025. URL https://gemini.google/overview/image-generation/.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*, 2025.

Kat Kampf and Nicole Brichtova. Experiment with gemini 2.0 flash native image generation, march 2025. *URL https://developers. googleblog. com/en/experiment-with-gemini-20-flash-native-image-generation/. Accessed*, 2025.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025a. URL https://arxiv.org/abs/2506.15742.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025b.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Arsh Koneru, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Reflect-DiT: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. *arXiv preprint arXiv:2503.12271*, 2025a. URL https://arxiv.org/abs/2503.12271.

Zongjian Li, Zheyuan Liu, Qihui Zhang, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Yang Ye, Wangbo Yu, Yuwei Niu, and Li Yuan. Uniworld-v2: Reinforce image editing with diffusion negative-aware finetuning and mllm implicit feedback. *arXiv preprint arXiv:2510.16888*, 2025b.

Bin Lin, Zongjian Li, Xinhua Cheng, Yuwei Niu, Yang Ye, Xianyi He, Shenghai Yuan, Wangbo Yu, Shaodong Wang, Yunyang Ge, et al. Uniworld: High-resolution semantic encoders for unified visual understanding and generation. *arXiv preprint arXiv:2506.03147*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025a.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025b.

Zheyuan Liu, Munan Ning, Qihui Zhang, Shuo Yang, Zhongrui Wang, Yiwei Yang, Xianzhe Xu, Yibing Song, Weihua Chen, Fan Wang, et al. Cot-lized diffusion: Let's reinforce t2i generation step-by-step. *arXiv preprint arXiv:2507.04451*, 2025c.

Xin Luo, Jiahao Wang, Chenyuan Wu, Shitao Xiao, Xiyan Jiang, Defu Lian, Jiajun Zhang, Dong Liu, et al. Editscore: Unlocking online rl for image editing via high-fidelity reward modeling. *arXiv preprint arXiv:2509.23909*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023. URL https://arxiv.org/abs/2303.17651.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6038–6047, 2023.

Yuwei Niu, Munan Ning, Mengren Zheng, Weiyang Jin, Bin Lin, Peng Jin, Jiaqi Liao, Chaoran Feng, Kunpeng Ning, Bin Zhu, et al. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2025.

OpenAI. Hello gpt-4o, May 2024. URL https://openai.com/index/hello-gpt-4o/.

OpenAI. Gpt-image-1. 2025. URL https://openai.com/index/introducing-4o-image-generation/.

OpenAI. Introducing GPT-4.1 in the API, April 2025a. URL https://openai.com/index/gpt-4-1/.

OpenAI. Image generation API. https://openai.com/index/image-generation-api/, 2025b.

Luozheng Qin, Jia Gong, Yuqing Sun, Tianjiao Li, Mengping Yang, Xiaomeng Yang, Chao Qu, Zhiyu Tan, and Hao Li. Uni-cot: Towards unified chain-of-thought reasoning across text and vision. *arXiv preprint arXiv:2508.05606*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

stepfun. Step-3o-vision: Stepfun's ai chat assistant. https://www.stepfun.com/chats/new/, 2025. Accessed: 2025-07-24.

Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative multimodal models are in-context learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14398–14409, 2024.

Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, et al. Longcat-image technical report. *arXiv preprint arXiv:2512.07584*, 2025.

Guo-Hua Wang, Shanshan Zhao, Xinjie Zhang, Liangfu Cao, Pengxin Zhan, Lunhao Duan, Shiyin Lu, Minghao Fu, Xiaohao Chen, Jianshan Zhao, et al. Ovis-u1 technical report. *arXiv preprint arXiv:2506.23044*, 2025a.

Linqing Wang, Ximing Xing, Yiji Cheng, Zhiyuan Zhao, Donghao Li, Tiankai Hang, Jiale Tao, Qixun Wang, Ruihuang Li, Comi Chen, et al. Promptenhancer: A simple approach to enhance text-to-image models via chain-of-thought prompt rewriting. *arXiv preprint arXiv:2509.04545*, 2025b.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. GenArtist: Multimodal LLM as an agent for unified image generation and editing. *arXiv preprint arXiv:2407.05600*, 2024. URL https://arxiv.org/abs/2407.05600.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025b.

Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, et al. Omnigen2: Exploration to advanced multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025c.

Haoning Wu, Zicheng Zhang, Erli Zhang, Chaofeng Chen, Liang Liao, Annan Wang, Chunyi Li, Wenxiu Sun, Qiong Yan, Guangtao Zhai, et al. Q-bench: A benchmark for general-purpose foundation models on low-level vision. In *ICLR*, 2024.

Keming Wu, Sicong Jiang, Max Ku, Ping Nie, Minghao Liu, and Wenhu Chen. Editreward: A human-aligned reward model for instruction-guided image editing. *arXiv preprint arXiv:2509.26346*, 2025d.

Yongliang Wu, Zonghui Li, Xinting Hu, Xinyu Ye, Xianfang Zeng, Gang Yu, Wenbo Zhu, Bernt Schiele, Ming-Hsuan Yang, and Xu Yang. Kris-bench: Benchmarking next-level intelligent image editing models. *arXiv preprint arXiv:2505.16707*, 2025e.

Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. Omnigen: Unified image generation. In *CVPR*, 2025.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Wang, Weiyun Ye, Shihao Geng, Yiren Zhao, Jiaming Li, Cunjian Li, Hang Sun, et al. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, 2023.

Zhiyuan Yan, Junyan Ye, Weijia Li, Zilong Huang, Shenghai Yuan, Xiangyang He, Kaiqing Lin, Jun He, Conghui He, and Li Yuan. Gpt-imgeval: A comprehensive benchmark for diagnosing gpt4o in image generation. *arXiv preprint arXiv:2504.02782*, 2025.

Ling Yang, Zhaochen Yu, Chenlin Meng, Minkai Xu, Stefano Ermon, and Bin Cui. Mastering text-to-image diffusion: Recaptioning, planning, and generating with multimodal llms. In *Forty-first International Conference on Machine Learning*, 2024.

Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2Img: Iterative self-refinement with GPT-4V(ision) for automatic image design and generation. *arXiv preprint arXiv:2310.08541*, 2023. URL https://arxiv.org/abs/2310.08541.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. Imgedit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.

Fukun Yin, Shiyu Liu, Yucheng Han, Zhibo Wang, Peng Xing, Rui Wang, Wei Cheng, Yingming Wang, Aojie Li, Zixin Yin, Pengtao Chen, Xiangyu Zhang, Daxin Jiang, Xianfang Zeng, and Gang Yu. ReasonEdit: Towards reasoning-enhanced image editing models. *arXiv preprint arXiv:2511.22625*, 2025. URL https://arxiv.org/abs/2511.22625.

Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. Anyedit: Mastering unified high-quality image editing for any idea. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26125–26135, 2025.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023a.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023b.

Qihui Zhang, Munan Ning, Zheyuan Liu, Yue Huang, Shuo Yang, Yanbo Wang, Jiayi Ye, Xiao Chen, Yibing Song, and Li Yuan. Upme: An unsupervised peer review framework for multimodal large language model evaluation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9165–9174, 2025a.

Yinan Zhang, Eric Tzeng, Yilun Du, and Dmitry Kislyuk. Large-scale reinforcement learning for diffusion models. In *European Conference on Computer Vision*, pp. 1–17. Springer, 2024.

Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. *arXiv preprint arXiv:2504.20690*, 2025b.

Haozhe Zhao, Xiaojian Shawn Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. Ultraedit: Instruction-based fine-grained image editing at scale. *Advances in Neural Information Processing Systems*, 37:3058–3093, 2024.

Xiangyu Zhao, Peiyuan Zhang, Kexian Tang, Xiaorong Zhu, Hao Li, Wenhao Chai, Zicheng Zhang, Renqiu Xia, Guangtao Zhai, Junchi Yan, et al. Envisioning beyond the pixels: Benchmarking reasoning-informed visual editing. *arXiv preprint arXiv:2504.02826*, 2025.

Le Zhuo, Liangbing Zhao, Sayak Paul, Yue Liao, Renrui Zhang, Yi Xin, Peng Gao, Mohamed Elhoseiny, and Hongsheng Li. From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. *arXiv preprint arXiv:2504.16080*, 2025. URL https://arxiv.org/abs/2504.16080.

APPENDIX CONTENTS

A Related Work **16**

A.1 Image Editing 16

A.2 Iterative Reasoning and Refinement 16

A.3 Reward Models for Image Editing 16

B Additional Evaluation on KRIS-Bench **18**

C Additional Ablation Studies **19**

C.1 Comparison with ReasonEdit 19

C.2 Reward-Weight Sensitivity 19

C.3 Instruction Enhancement Paradigms 20

C.4 Different EditThinker Experts 20

C.5 Joint Scoring by the Expert 20

D Additional Implementation Details **21**

D.1 Details of EditThinker Expert 21

D.2 Details of EditThinker 21

D.3 Hyperparameters and Stopping Criterion 21

E Details of THINKEDIT-167K Dataset **21**

F Visualization **22**

G Qualitative Analysis **28**

A Related Work

A.1 Image Editing

The emergence of diffusion models marked a paradigm shift in Text-to-Image (T2I) synthesis (Song et al., 2020; Rombach et al., 2022; Lipman et al., 2022; Liu et al., 2025c; Yan et al., 2025). Image editing, however, imposes stricter constraints to balance attribute modification with background preservation. Early solutions, ranging from inversion-based techniques (Meng et al., 2021; Hertz et al., 2022; Mokady et al., 2023) to explicit spatial controls (Zhang et al., 2023b; Ye et al., 2023), improved precision but often suffered from computational overhead or limited semantic flexibility. While initial instruction-tuning attempts (Brooks et al., 2023) introduced natural language control, they faced generalization bottlenecks. Recently, the field has advanced towards robust instruction-tuned models (Zhang et al., 2025b; Liu et al., 2025a; Batifol et al., 2025) and general-purpose multimodal LLMs or unified models (Wu et al., 2025b; OpenAI, 2025b; Wu et al., 2025c; Li et al., 2025b), evolving alongside foundational architectures like flow matching (Labs et al., 2025a). Although the foundational capabilities of editing models continue to improve, single-turn editing can still fail to follow complex instructions. EditThinker addresses these failures through a learned instruction-refinement module that can be applied to multiple existing editors without updating their parameters.

A.2 Iterative Reasoning and Refinement

Iterative feedback and refinement have been studied extensively in language models. Self-Refine (Madaan et al., 2023) uses the same LLM to generate an initial answer, provide feedback, and revise the answer without additional model training. Reflexion (Shinn et al., 2023) converts task feedback into verbal reflections stored in episodic memory to guide subsequent trials, also without updating model weights. These methods establish the usefulness of iterative refinement; our contribution concerns learning an editing-specific reasoning module within this broader family of approaches.

In visual generation, Idea2Img (Yang et al., 2023) uses GPT-4V to revise T2I prompts from feedback on generated images. GenArtist (Wang et al., 2024) supports both image generation and editing: an MLLM agent decomposes requests, selects tools, and performs verification and correction through a planning tree. These systems demonstrate how inference-time planning and feedback can coordinate existing visual models. Reflect-DiT (Li et al., 2025a) trains a VLM judge and adapts a diffusion transformer to condition on previous generations and textual feedback. ReflectionFlow (Zhuo et al., 2025) fine-tunes a visual reflector and a diffusion-based corrector for T2I refinement, combining reflection with noise- and prompt-level search. Both approaches adapt the visual generation component to consume reflection, rather than keeping an existing editor unchanged.

ReasonEdit (Yin et al., 2025) directly targets image editing through a thinking-editing-reflection loop. Its multi-stage training first adapts the MLLM reasoner and diffusion generator separately and then jointly fine-tunes both components. EditThinker instead uses SFT and RL to train a plug-and-play VLM for critique and instruction refinement while keeping the underlying editors fixed. Its learned reasoning module can therefore be evaluated with different existing editors without retraining those editors. The reward design combines format adherence, critique agreement with expert scores of the current edit, and subsequent editing improvement. We study how these training signals affect the quality-efficiency trade-off during iterative editing.

A.3 Reward Models for Image Editing

Feedback and Reward Modeling in Image Editing. The correlation between Multimodal Large Language Models (MLLMs) and human perception (Chen et al., 2024; Zhang et al., 2025a) has established the "MLLM-as-a-Judge" paradigm, facilitating their use as reward models (RMs) for generative tasks (Xu et al., 2023; Niu et al., 2025). However, translating holistic evaluations into effective training signals for image editing is non-trivial. Early attempts using discrete scores (Gong et al., 2025) or dense logit-based values (Wu et al., 2024) often failed to capture the fine-grained nuances required for precise visual modifications. To address these limitations, recent research has pivoted towards domain-specialized reward modeling. Notably, EditReward (Wu et al., 2025d) constructed a large-scale human preference dataset to train a reward model capable of rigorous data filtering

and alignment. Building on this, EditScore (Luo et al., 2025) developed a series of specialized reward models that surpass general-purpose VLM judges, successfully unlocking effective online reinforcement learning (RL) for editing policies.

Despite these advancements in RL application, a fundamental "feedback lag" remains. Existing specialized RMs primarily provide outcome-oriented feedback—they evaluate the edited image after generation. This post-hoc signal acts as an external judge rather than an internal guide. In complex editing scenarios requiring multi-step reasoning, such scalar rewards fail to correct the intermediate logic of the generation process (Zhang et al., 2024). Consequently, an emerging paradigm seeks to utilize MLLMs not merely as judges, but as internal planners (Liu et al., 2025c; Yang et al., 2024). In this work, we shift from maximizing a static post-hoc reward to harnessing the MLLM's structured reasoning process to actively guide the editing model during execution.

B ADDITIONAL EVALUATION ON KRIS-BENCH

We provide the full KRIS-Bench results in table 5. KRIS-Bench (Wu et al., 2025e) evaluates factual, conceptual, and procedural knowledge in image editing. As specified in the table caption, the Temporal subset is excluded from our comparison because some baselines do not support multi-image inputs.

Table 5: Comparison of model performance on Kris-Bench. ‡ indicates results from our own tests with official model checkpoint. 0.0* **indicates that the model was not evaluated on multi-image editing**. Since some baseline currently does not support multi-image inputs, we excluded the Temporal subset to ensure a fair comparison.

Model	Factual Knowledge				Conceptual Knowledge			Procedural Knowledge			Overall Score
	Attribute	Spatial	Temporal	Average	Social Sci.	Natural Sci.	Average	Logical	Instruction	Average	
Proprietary Models											
Doubao (ByteDance, 2025a)	70.92	59.17	40.58	63.30	65.50	61.19	62.23	47.75	60.58	54.17	60.70
Step 3o vision (stepfun, 2025)	69.67	61.08	63.25	66.70	66.88	60.88	62.32	49.06	54.92	51.99	61.43
Gemini 2.0 (Kampf & Brichtova, 2025)	66.33	63.33	63.92	65.26	68.19	56.94	59.65	54.13	71.67	62.90	62.41
GPT-4o (OpenAI, 2024)	83.17	79.08	68.25	79.80	85.50	80.06	81.37	71.56	85.08	78.32	80.09
Open-source Models											
InstructPix2Pix (Brooks et al., 2023)	30.33	21.33	0.00*	23.33	22.56	26.56	25.59	19.81	14.75	17.28	22.82
OmniGen (Xiao et al., 2025)	37.92	28.25	21.83	33.11	30.63	27.19	28.02	11.94	35.83	23.89	28.85
MagicBrush (Zhang et al., 2023a)	53.92	39.58	0.00*	41.84	42.94	38.06	39.24	30.00*	23.08	26.54	37.15
AnyEdit (Yu et al., 2025)	47.67	45.17	0.00*	39.26	38.56	42.94	41.88	36.56	26.92	31.74	38.55
Emu2 (Sun et al., 2024)	51.50	48.83	22.17	45.40	34.69	38.44	37.54	24.81	45.00	34.91	39.70
Step1X-Edit (Liu et al., 2025b)	55.50	51.75	0.00*	45.52	44.69	49.06	48.01	40.88	22.75	31.82	43.29
HiDream-E1 (Cai et al., 2025)	52.75	49.42	0.00*	43.31	52.56	49.25	50.05	45.19	30.08	37.64	44.72
ByteMorph (Chang et al., 2025)	61.17	62.00	0.00*	51.27	45.50	47.38	46.92	32.00	31.33	31.67	44.85
FLUX.1 Kontext [Dev] (Batifol et al., 2025)	64.83	60.92	0.00*	53.28	48.94	50.81	50.36	46.06	39.00	42.53	49.54
OmniGen2 (Wu et al., 2025c)	59.92	52.25	54.75	57.36	47.56	43.12	44.20	32.50	63.08	47.79	49.71
UniWorld-V1 (Lin et al., 2025)	58.17	54.50	63.00	47.71	47.50	43.94	44.80	42.00	53.83	47.92	50.27
Step1X-Edit v1.1 (Liu et al., 2025b)	64.17	61.75	0.00*	53.05	52.06	55.06	54.34	52.56	36.75	44.66	51.59
BAGEL (Deng et al., 2025)	64.27	62.42	42.45	60.26	55.40	56.01	55.86	52.54	50.56	51.69	56.21
BAGEL-Think (Deng et al., 2025)	67.42	68.33	58.67	66.18	63.55	61.40	61.92	48.12	50.22	49.02	60.18
Uni-CoT (Qin et al., 2025)	72.76	72.87	67.10	71.85	70.81	66.00	67.16	53.43	73.93	63.68	68.00
Critique-Refine-Repeat											
OmniGen2‡	60.21	54.67	0.00*	58.73	53.60	46.76	48.42	37.67	56.67	45.84	50.52
+ EditThinker-8B	62.18	53.92	0.00*	59.98	61.50	49.90	52.71	37.04	58.78	46.43	**53.09**
+ EditThinker-Expert-GPT4.1	65.55	56.83	0.00*	63.22	63.35	55.26	57.22	44.38	60.44	51.26	57.34
FLUX.1 Kontext [Dev]‡	71.12	67.25	0.00*	70.09	56.60	58.91	58.35	56.75	63.72	59.75	61.81
+ EditThinker-8B	77.82	65.50	0.00*	73.73	73.44	69.04	70.09	62.29	65.33	63.60	**69.53**
+ EditThinker-Expert-GPT4.1	81.03	74.67	0.00*	79.33	77.60	74.98	75.62	71.38	65.50	68.86	74.93
Qwen-Image-Edit ‡	72.73	73.33	0.00*	72.89	63.50	60.40	61.15	57.47	67.97	61.70	64.43
+ EditThinker-8B	78.48	73.83	0.00*	77.24	76.20	70.69	72.02	65.23	66.89	65.94	**71.91**
+ EditThinker-Expert-GPT4.1	83.70	76.08	0.00*	81.67	81.99	80.53	80.91	71.94	76.07	73.40	79.34
Longcat-Image-Edit ‡	77.67	75.25	0.00*	77.02	65.50	63.12	63.69	61.79	69.00	64.88	67.52
+ EditThinker-8B	82.61	80.33	0.00*	82.00	77.30	73.31	74.28	68.04	70.11	68.93	**75.00**
+ EditThinker-Expert-GPT4.1	85.58	81.67	0.00*	84.53	83.00	80.44	81.06	71.04	70.11	70.64	79.40

Performance on Reasoning Editing. On KRIS-Bench, which evaluates factual and conceptual knowledge, our method demonstrates strong performance on reasoning-driven edits. As shown in table 5, we observe consistent performance gains across all models, with four image edit models all experiencing notable improvements in their overall scores.

C Additional Ablation Studies

This section reports additional reference comparisons, reward-weight ablations, and prompt-enhancement and expert-model comparisons. The thinker-design, data-partition, VLM-backbone, and alternative-judge experiments are presented in the main paper (tables 3c, 3d and 4). Where efficiency is reported, *average rounds* denotes the mean number of editing iterations. It serves as a proxy for editor calls and does not include a new measurement of wall-clock latency or the cost of individual VLM calls.

C.1 Comparison with ReasonEdit

Table 6 compares Qwen-Image-Edit-2509 with ReasonEdit (Yin et al., 2025) and EditThinker-8B. Our reproduced baseline scores 7.51 on GEdit-Bench-EN and 4.24 on ImgEdit-Bench. Adding EditThinker-8B raises these scores to 7.85 and 4.58; the published ReasonEdit results are 7.77 and 4.36. ReasonEdit adapts the editor, whereas EditThinker keeps its weights fixed. These results provide a reference comparison across the respective evaluation settings; they do not isolate the thinker's contribution under a matched adaptation and inference budget.

Table 6: Comparison on GEdit-Bench-EN and ImgEdit-Bench with Qwen-Image-Edit-2509 as the base editor. The baseline and EditThinker-8B results are from our evaluation; ReasonEdit results are taken from the published paper (Yin et al., 2025). Higher scores are better.

Method	GEdit-Bench-EN	ImgEdit-Bench
Qwen-Image-Edit-2509	7.51	4.24
+ ReasonEdit	7.77	4.36
+ EditThinker-8B	7.85	4.58

C.2 Reward-Weight Sensitivity

Table 7 varies the weights of the format, critic, and edit rewards. The selected setting $(\alpha, \beta, \gamma) = (0.50, 0.10, 0.40)$ yields a GEdit-Bench-EN overall score of 7.02 at 2.19 average rounds and an ImgEdit-Bench score of 4.07 at 1.85 average rounds. Removing the critic weight while increasing the edit weight to 0.50 yields a higher ImgEdit-Bench score of 4.15, with 3.70 average rounds; its GEdit-Bench-EN score is 6.95 at 3.93 rounds. These measurements illustrate a quality-iteration trade-off among the tested settings.

The setting with $\alpha = 0.25$ has lower scores on both benchmarks, but it also changes β and γ. This comparison does not isolate the effect of the format weight or establish format forgetting as the cause. Similarly, changing the critic weight in the $\alpha = 0.50$ sweep also changes the edit weight. The table supports comparisons between complete reward configurations rather than independent causal effects of each coefficient.

Table 7: Reward-weight ablation. Weights (α, β, γ) apply to the format, critic, and edit rewards, respectively. Avg. rounds is the reported average number of editing iterations, used as an editor-call proxy rather than a wall-clock measurement.

Reward weights			GEdit-Bench-EN				ImgEdit-Bench	
α	β	γ	$G_{\mathrm{SC}}\uparrow$	$G_{\mathrm{PQ}}\uparrow$	$G_{\mathrm{O}}\uparrow$	Avg. rounds↓	Overall↑	Avg. rounds↓
0.50	0.10	0.40	7.59	7.63	7.02	2.19	4.07	1.85
0.25	0.15	0.60	7.22	7.59	6.70	2.03	3.84	1.98
0.50	0.00	0.50	7.43	7.65	6.95	3.93	4.15	3.70
0.50	0.20	0.30	7.44	7.64	6.96	2.16	4.05	1.89
0.50	0.30	0.20	7.40	7.55	6.88	2.13	4.06	1.94
0.50	0.40	0.10	7.39	7.54	6.87	2.14	4.02	1.96
0.50	0.50	0.00	7.34	7.50	6.74	2.13	4.01	1.94

C.3 Instruction Enhancement Paradigms

In table 8a, EditThinker-8B reaches a GEdit overall score of 7.02, exceeding PromptEnhancer-32B (6.53) and GPT-4.1 prompt enhancement (6.82), while remaining below the GPT-4.1 EditThinker expert (7.19). Combining prompt enhancement and iterative refinement with GPT-4.1 yields 7.06, below iterative refinement alone. This suggests that prompt enhancement and iterative feedback are not automatically additive; the table does not isolate the cause of this degradation.

C.4 Different EditThinker Experts

In table 8b, GPT-4.1 achieves the highest GEdit overall score among the tested experts (7.19), followed by Gemini-2.5-Pro (7.11), GPT-4o (7.05), and Doubao-1.5 (6.80). The results show that the loop can use different expert VLMs; they do not establish equal computational cost across these models.

Table 8: Instruction enhancement and expert-model comparisons on GEdit-Bench-EN with FLUX.1-Kontext [Dev]. All scores are higher-is-better.

(a) Instruction enhancement paradigms.

Method	GEdit-Bench-EN ↑		
	G_SC	G_PQ	G_O
FLUX.1-Kontext [Dev]	6.62	7.61	6.18
+ PromptEnhancer-32B	6.94	7.62	6.53
+ PromptEnhancer w. GPT 4.1	7.34	7.64	6.82
+ EditThinker-8B	7.59	7.63	7.02
+ EditThinker w. GPT 4.1	7.83	7.66	7.19
+ Both w. GPT 4.1	7.75	7.60	7.06

(b) Expert VLMs in Critique-Refine-Repeat.

Model	GEdit-Bench-EN ↑		
	G_SC	G_PQ	G_O
FLUX.1-Kontext [Dev]	6.62	7.61	6.18
+ GPT 4.1	7.83	7.66	7.19
+ Gemini 2.5 Pro	7.76	7.65	7.11
+ Doubao 1.5	7.36	7.59	6.80
+ GPT-4o	7.65	7.67	7.05

C.5 Joint Scoring by the Expert

Table 9 reports a separate experiment in which the expert predicts scores while refining instructions. Relative to expert refinement alone, the GEdit-Bench-EN score decreases from 7.19 to 6.86 and the ImgEdit-Bench score from 4.13 to 3.89. These results motivate separating scoring from instruction refinement during expert data generation; they do not establish task complexity or domain mismatch as the cause of the decrease.

Table 9: Expert refinement with and without jointly predicting edit scores. The reported comparison uses the GPT-4.1 expert; higher benchmark scores are better.

Expert task	GEdit-Bench-EN	ImgEdit-Bench
Refinement	7.19	4.13
Joint scoring and refinement	6.86	3.89

D Additional Implementation Details

D.1 Details of EditThinker Expert

To supervise EditThinker with high-quality reasoning traces and refined editing instructions, we employ GPT-4.1 as the *expert* model. At the t-th editing iteration, we provide the tuple $(I_{\text{src}}, I_{\text{edit}}^{t-1}, T_s, T_{t-1})$ as input to the expert. The model then generates a reasoning trace R_t, a refined editing instruction T_t, and a <stop> flag indicating whether the current edit successfully satisfies the user's intent. The expert prompt is meticulously designed to explicitly encourage a multi-step cycle. It requires the model to: (1) evaluate whether the edited image fulfills the original instruction T_s, (2) identify failure causes through detailed reasoning, and (3) synthesize an improved instruction that corrects these errors without introducing new inconsistencies. The full prompt template used for the expert is provided in fig. 6.

D.2 Details of EditThinker

EditThinker adopts a unified prompt format for both training and inference. This design ensures that the behavior learned during supervision aligns seamlessly with the capabilities required at inference time, enabling the model to (1) evaluate the current result, (2) reason about potential issues, and (3) refine the instruction for the next round.

At each iteration t, EditThinker receives a multimodal tuple $(I_{\text{src}}, I_{\text{edit}}^{t-1}, T_s, T_{t-1})$ that provides the complete context of the editing state. Here, I_{src} and T_s represent the original source image and user instruction; I_{edit}^{t-1} denotes the intermediate result from the previous turn; and T_{t-1} is the specific instruction that produced it. Based on this input, EditThinker outputs three components: semantic and quality scores S_t, a natural-language reasoning trace R_t, and a refined editing instruction T_t. Diverging from prior systems that rely on a binary <stop> flag, we implement a continuous scoring scheme comprising a *Semantic Score* and a *Quality Score*. This offers two key advantages: (1) it provides a smoother, more informative supervision signal for learning nuanced failure patterns; and (2) it enables precise control over inference quality, allowing users to trigger refinement only when the predicted score falls below a specific threshold. The full prompt template used for the EditThinker is provided in fig. 7.

D.3 Hyperparameters and Stopping Criterion

We use reward weights $(\alpha, \beta, \gamma) = (0.5, 0.1, 0.4)$ for format, critique, and edit rewards, respectively. The trajectory score-variance threshold is $\theta = 3.0$. At inference, early stopping uses the geometric mean of the two predicted scores:

$$S_{\text{stop},t} = \sqrt{S_{\text{sem},t}\, S_{\text{qual},t}}, \qquad \text{stop if } S_{\text{stop},t} \geq 8. \tag{6}$$

The default maximum is $N = 5$ editing rounds. This inference budget is distinct from the GRPO rollout group size of 8 used during training. Unless an ablation varies the round budget, these are the default inference settings. Reward-weight ablations appear in Appendix C, and design ablations in table 3c.

E Details of ThinkEdit-167K Dataset

We obtain T_s and I_{src} from three data sources: OpenGPT-4o-Image, ShareGPT-4o-Image, and Pico-Banana-400K. From these sources, we sample 40K, 40K, and 60K editing instances respectively, ensuring that the editing categories are as evenly distributed as possible, resulting in a total of 140K raw samples. We divide these samples into three splits and use GPT-4.1 as the EditThinker-Expert, while selecting OmniGen2, FLUX.1 Kontext [Dev], and Qwen-Image-Edit as the editors. After trajectory filtering, we retain 70K valid trajectories. Among them, 10K trajectories are selected for RL, while the remaining 60K undergo step-wise refinement and filtering, ultimately producing 140K high-quality samples for SFT. Unrolling the 10K selected RL trajectories yields 27K step-wise samples for RL training.

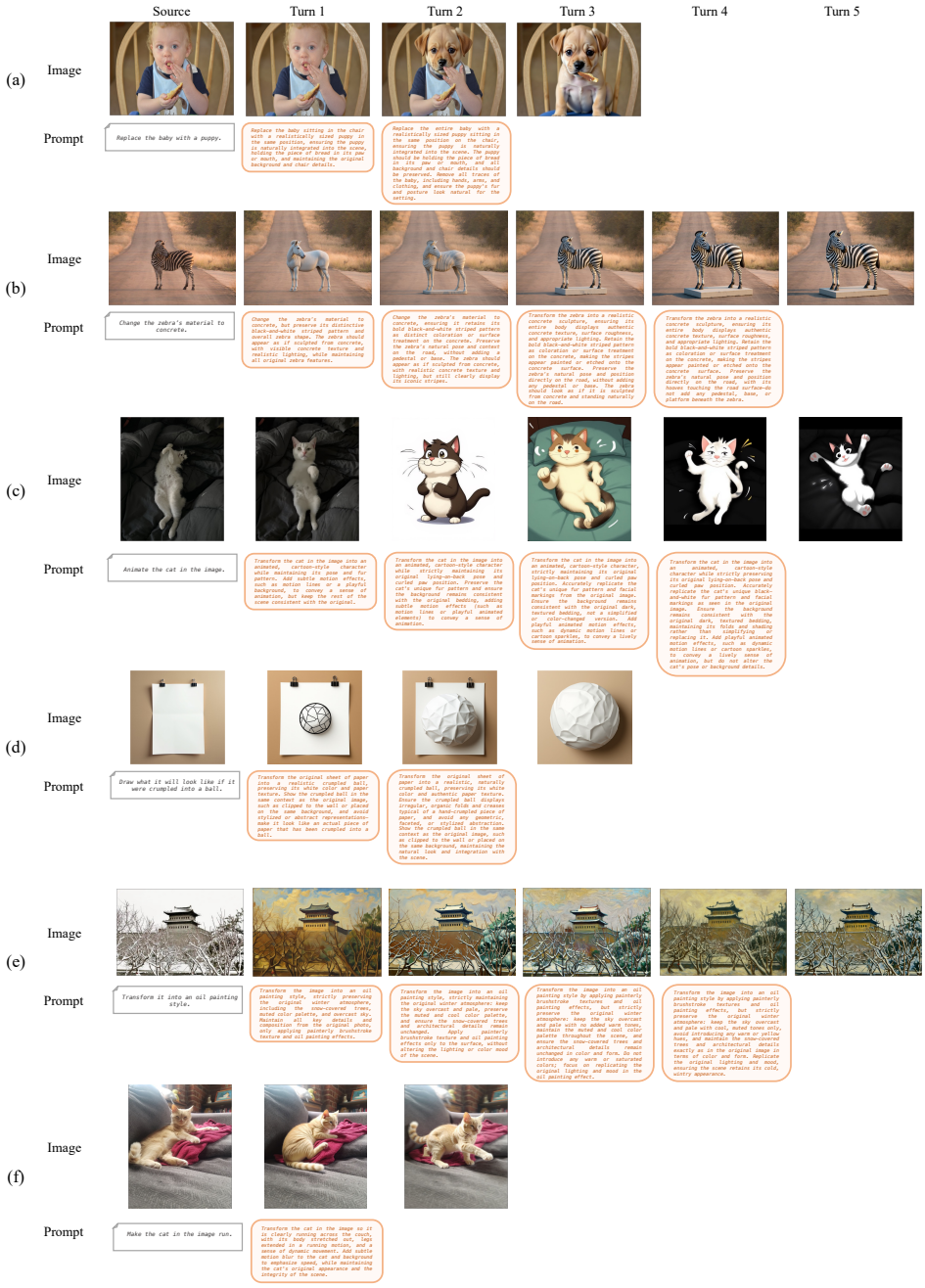

Figure 4: Qualitative visualizations of EditThinker paired with different editors. Subfigures (a) and (b) show results with FLUX.1 Kontext [Dev], (c) and (d) use OmniGen2, and (e) and (f) use Qwen-Image-Edit.

F VISUALIZATION

We provide visualizations of the outputs generated by our framework across different settings. As shown in fig. 4, EditThinker produces high-quality edits when combined with various editors, including FLUX.1 Kontext [Dev], OmniGen2, and Qwen-Image-Edit. In addition, fig. 5 visualizes EditThinker's iterative reasoning dynamics with FLUX.1 Kontext [Dev], highlighting how the Thinker critiques intermediate results and progressively refines the instruction over multiple rounds.



Figure 5: Visualization of EditThinker's reasoning traces and intermediate editing results when paired with FLUX.1 Kontext [Dev]. The figure illustrates how the Thinker evaluates the current output, identifies issues, and iteratively refines the instruction over multiple rounds.

You are an expert image editing evaluator and prompt engineer. Your task is to:

1. Evaluate whether an edited image successfully fulfills the original user instruction.
2. If not satisfied, generate an improved rewritten prompt that addresses the shortcomings.

Input Information. You will receive:

- **Original Image**: the input image before editing.
- **Original User Instruction**: the user's initial editing request.
- **Rewritten Prompt**: the refined instruction that was used for editing.
- **Edited Image**: the resulting image after applying the rewritten prompt.

Evaluation Criteria.
A. Intent Alignment

- Does the edited image achieve the core goal of the user instruction?
- Are all requested changes present and correctly implemented?

B. Quality Assessment

- **Subject/Object Changes**: correctness of additions/removals/replacements.
- **Appearance Modifications**: accuracy of color/style/material edits.
- **Scene Changes**: correctness of background/environment edits.
- **Detail Preservation**: important details remain intact.
- **Visual Coherence**: edited image looks natural and well-integrated.

C. Common Failure Patterns

- Missing requested elements.
- Incorrect positioning or scale.
- Wrong colors or materials.
- Unnatural blending or artifacts.
- Lost subject details.
- Style inconsistency.
- Text errors (if applicable).
- Over-editing or under-editing.

Evaluation Decision.
SATISFIED: the edited image fulfills the original instruction with acceptable quality. Minor imperfections are acceptable if the core intent is achieved.
NOT SATISFIED: the edit fails in key aspects. Major elements missing, incorrect, or severe quality issues $\rightarrow$ refinement required.

Prompt Refinement Strategy (If Not Satisfied).
1. Identify what went wrong.

- Compare original instruction $\rightarrow$ rewritten prompt $\rightarrow$ edited result.
- Identify mismatches between intent and execution.
- Determine whether the issue is clarity, specificity, or contradiction.

2. Refinement approaches.

- **If vague:** add specific descriptors, spatial relations, or context.
- **If contradictory:** resolve conflicts and simplify.
- **If important details were lost:** explicitly require preservation.
- **If scale/position wrong:** add precise location and size cues.
- **If style incorrect:** specify textures, lighting, materials.
- **If over/under-edited:** specify degree of modification.

3. Leverage all information.

- Reference visible content in the original and edited images.
- Retain what worked; correct what failed.

Output Format.

```
{
"is_satisfied": true/false,
"reason": "Detailed explanation of evaluation. If satisfied,
explain why it meets requirements. If not satisfied, describe
specific shortcomings.",
"new_rewritten_prompt": "Only include if is_satisfied is false.
If satisfied, set to null."
}
```

Examples.
Example 1: Satisfied

```
{ "is_satisfied": true,
"reason": "The edited image successfully adds a cat...",
"new_rewritten_prompt": null }
```

Example 2: Not Satisfied — Lack of Specificity

```
{ "is_satisfied": false,
"reason": "The rewritten prompt was too vague...",
"new_rewritten_prompt": "Change the car color to blue..." }
```

Input Data:

- **Original User Instruction**: T_s
- **Rewritten Prompt Used**: T_{t-1}
- **Images Order**: [Original Image, Edited Image]

Images: I_{src}, I_{edit}^{t-1}

Figure 6: **EditThinker Expert Prompt.** The full expert instruction used for EditThinker Expert. At each iteration, the Expert observes $(I_{src}, I_{edit}^{t-1}, T_s, T_{t-1})$ and produces stop flag, reasoning, and a refined instruction.

Images: I_{src}, I_{edit}^{t-1}

Edit Evaluation and Prompt Refinement System.
You are an expert image editing evaluator and prompt engineer. Your task is to:

1. Score the edited image from two perspectives and output the result in JSON format.
2. If you think the edited image is not good enough, generate an optimized rewritten prompt that addresses the original shortcomings; if you think it is good enough, output the original rewritten prompt.

Input Information. You are shown two images in sequence:

- **Original Image**: the input image before editing.
- **Edited Image**: the latest edited image generated using the previous instruction.

The textual instructions involved in this process are:

- **Original User Instruction** : "T_s"
- **Previous Rewritten Instruction** : "T_{t-1}"

Evaluation Criteria (Score 0–10).
A. Semantic Score (Instruction Following and Preservation).

- Evaluates how accurately the edit was performed. The edit fails if it either (A) fails to follow the text instruction, or (B) over-edits the image by changing content that was not supposed to be changed.
- (10) Follows the instruction perfectly *and* preserves all unchanged content.
- (0) Fails to follow the instruction *or* needlessly changes the original scene.

B. Quality Score (Naturalness & Artifacts).

- Evaluates the technical quality of the newly edited image.
- (10) Looks natural, has no artifacts, and integrates seamlessly.
- (0) Looks unnatural (wrong shadow, lighting, or sense of distance) or contains severe artifacts (distortion, blurred faces, unusual body parts, disharmony).

Prompt Refinement Strategy (if Not Good Enough). When generating a new rewritten prompt, follow these steps:
1. What went wrong?

- Compare original instruction → rewritten prompt → edited result.
- Identify gaps between intent and execution.
- Determine if the issue is clarity, specificity, or contradiction.

2. Refinement approaches.

- **If the rewritten prompt was too vague:** add more specific descriptors (exact colors, positions, sizes), include spatial relationships and context, and specify interaction with existing elements.
- **If the rewritten prompt was contradictory:** resolve conflicts between requirements, prioritize core intent over secondary details, and simplify complex multi-part instructions.
- **If important details were lost:** explicitly state preservation requirements, add "maintain [aspect]" or "preserve [feature]" clauses, and reference specific elements from the original image.
- **If positioning/scale was wrong:** use more precise spatial descriptors, add relative size/scale indicators, and specify foreground/midground/background placement.
- **If style/appearance was incorrect:** use more specific visual vocabulary, add reference to the original image's style elements, and include material/texture/lighting specifications.
- **If the edit was over/under-processed:** add modifiers such as "subtle", "gentle", "dramatic", or "significant", specify the degree of change more clearly, and balance enhancement with naturalness.

3. Leverage all information.

- Reference what is visible in the original image.
- Learn from what the previous rewritten prompt missed.
- Use the edited image as feedback on what went wrong.
- Maintain what worked, and only modify what needs to be improved.

Output Format. The output consists of three parts:

1. A statement: analysis process and reasoning.
2. A JSON object: scores in different dimensions.

3. A prompt: either the optimized rewritten prompt or the original rewritten prompt.

An example output is shown below:

```
<think>
Detailed explanation of evaluation and new rewritten prompt.  If the
edited image is good enough, explain why it meets the requirements.
If it is not good enough, explain the specific shortcomings.
</think>
<score>
{
"semantic":  [0-10],
"quality":  [0-10]
}
</score>
<answer>
"Improved rewritten prompt that addresses the identified issues and
enhances clarity, specificity, and preservation requirements" (if
NOT GOOD ENOUGH)
"Original rewritten prompt" (if GOOD ENOUGH)
</answer>
```

Figure 7: **EditThinker prompt.** The unified prompt template used for EditThinker's edit evaluation and instruction refinement. At each iteration, the Thinker observes $(I_{src}, I_{edit}^{t-1}, T_s, T_{t-1})$ and produces semantic and quality scores, reasoning, and a refined instruction.

G Qualitative Analysis

Figure 8 shows background-editing failures and long sequences with FLUX.1-Kontext-dev and EditThinker-8B. In panel (a), the request changes the lawn to a beach and replaces a tree with an umbrella, but later edits remove the phone booth; the displayed GEdit score falls from 4.33 to 3.67. In panel (b), further refinement of a dog-on-a-beach edit introduces additional changes, and the displayed score decreases from 4.0 to 3.33. These examples show that extra refinement can alter content beyond the requested background change.

Panels (c) and (d) show facial-expression and Chinese-text editing sequences extending to turn 16. The final facial-expression edit contains conspicuous eye-region artifacts, and the displayed Chinese text still does not match the requested replacement. These examples illustrate persistent errors despite additional iterations; the images alone do not separate limitations of the thinker from those of the editor.

Figure 9 contains two examples marked successful and one marked unsuccessful. Panel (a) assembles separate body parts into a running figure. Panel (b) changes the car-door image to show scratches, without retaining the key and hand that appear in the intermediate result. Panel (c) changes the bedroom viewpoint but remains labeled as failing the requested view from beside the bed toward the wardrobe. These selected examples illustrate both successful refinement and a remaining spatial-viewpoint failure; they do not provide a quantitative estimate of success on complex instructions.

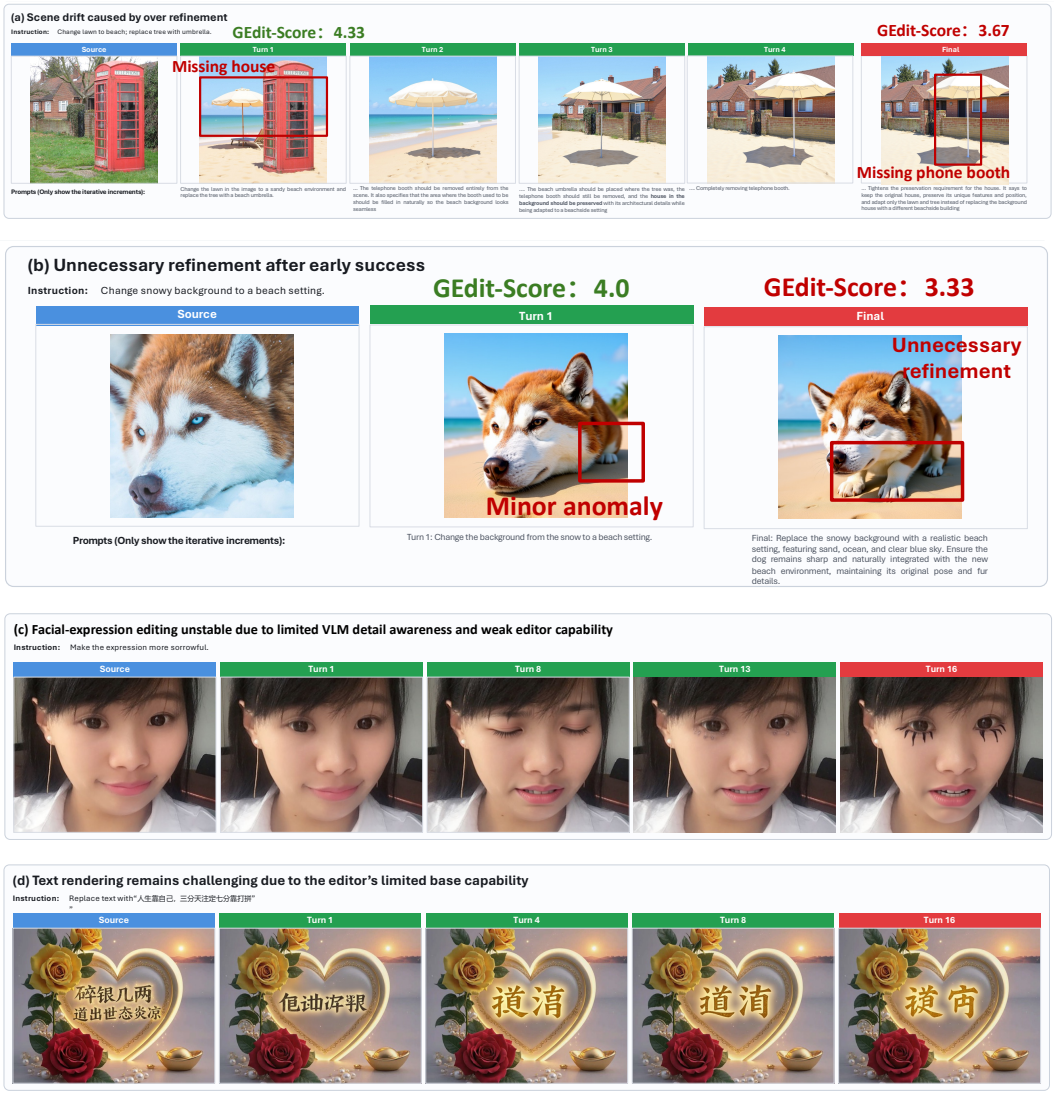

Figure 8: Failure and unresolved cases with FLUX.1-Kontext-dev and EditThinker-8B. Original panels (a)–(d) are arranged vertically: (a) background editing changes unrequested scene content; (b) further refinement degrades an early result; (c) facial-expression editing and (d) text replacement remain unresolved through the displayed turn 16. Scores and annotations are retained from the original examples.

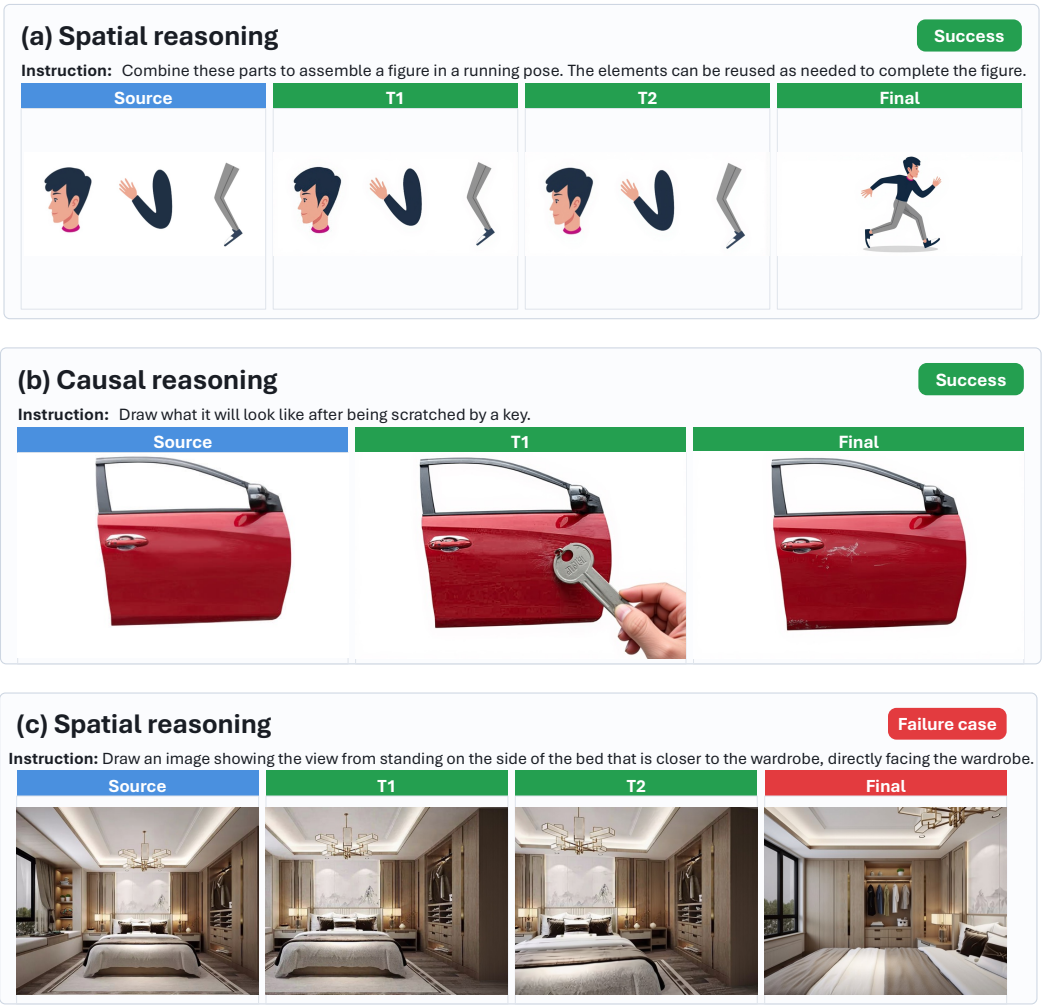

Figure 9: Challenging instructions with FLUX.1-Kontext-dev and EditThinker-8B. Original panels (a)–(c) are arranged vertically: assembling a running figure, depicting a scratched car door, and changing the bedroom viewpoint. The source examples mark the first two as successes and the third as a failure. Original instructions and intermediate results are preserved.