FEEDBACK-DRIVEN PROMPT REFINEMENT FOR IMAGE EDITING VIA SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Instruction-guided image editing enables users to modify images through natural-language instructions, yet editing quality remains sensitive to the prompts used to control the editor, creating a gap between user intent and model behavior. Existing iterative refinement methods often explicitly evaluate intermediate editing results before generating revised instructions, coupling evaluation and refinement within the deployed policy. Such intermediate assessments can misguide subsequent refinement when inaccurate, while also introducing additional supervision and inference requirements. We propose **FeedEdit**, a feedback-driven prompt-refinement framework that learns refinement decisions directly from editing outcomes. FeedEdit formulates prompt refinement as an adaptive stop-or-refine decision problem, where evaluator feedback is used during training to teach the policy whether to stop or how to refine, eliminating the need to explicitly generate an intermediate evaluation at inference. Our key insight is that outcome feedback can supervise the policy in two complementary ways: *causal feedback* evaluates the current edit and guides the construction of stopping and refinement targets for offline learning, while *hindsight feedback* evaluates the outcome of a policy-sampled action and provides token-level supervision for online self-distillation. On ImgEdit-Bench and GEdit-Bench-EN, FeedEdit achieves overall scores of 4.54 and 8.13 with FLUX.2-klein-9B, outperforming strong state-of-the-art refinement methods. Further analyses show that FeedEdit learns effective adaptive stop-or-refine behavior and transfers across editors without retraining. We plan to release the dataset, model checkpoints, and code.

1 INTRODUCTION

Instruction-guided image editing has emerged as an intuitive and flexible paradigm for controllable visual generation, enabling users to modify images through natural-language instructions (Brooks et al., 2023; Zhang et al., 2023). However, a persistent bottleneck lies in the quality of the prompts used to control editing models. Due to their sensitivity to textual inputs, semantically equivalent instructions can lead to substantially different editing outcomes. As a result, effective editing often requires carefully crafted prompts, creating a gap between user intent and model behavior.

Recent work attempts to bridge this gap by refining user instructions before or during the editing process. Methods based on editing plans and structured reasoning improve the prompt before execution (Zeng et al., 2026b; Zhou et al., 2026), while iterative approaches inspect intermediate results through critique, reflection, or correction (Li et al., 2025; Yin et al., 2026; Zeng et al., 2026a). Although iterative reasoning can adapt to errors revealed by the editor, such evaluate-then-refine methods couple explicit evaluation and prompt generation within the deployed policy. This coupling introduces two limitations. First, inaccurate assessments can mislead subsequent refinement and may even degrade otherwise correct outputs (Song et al., 2025; Tyen et al., 2024; Huang et al., 2024; Kamoi et al., 2024). Second, learning and decoding explicit assessments introduces additional supervision and inference requirements. This raises a natural question:

Can a refiner learn when and how to refine an editing instruction without explicitly generating an evaluation first?

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.

In this work, we propose **FeedEdit**, a feedback-driven prompt-refinement framework that learns refinement decisions directly from editing outcomes, rather than requiring the deployed policy to explicitly evaluate each result before deciding how to act. We formulate prompt refinement as an iterative decision problem, where the policy observes the current editing context and decides whether to stop or produce a refined instruction. During training, an external evaluator assesses the resulting edits and provides supervision for these decisions. Our key insight is that the outcome feedback can supervise the policy in two complementary ways depending on when it is observed: *causal feedback* evaluates the current edit and is used to construct stopping and refinement targets for offline learning, whereas *hindsight feedback* evaluates the outcome of a policy-sampled action and provides token-level supervision for online self-distillation. This allows FeedEdit to leverage explicit evaluator feedback during training, while eliminating the need for intermediate evaluation at inference.

This formulation has two practical advantages. First, FeedEdit makes refinement decisions directly from the editing context, without relying on a potentially error-prone intermediate assessment. Second, it removes the need to supervise and decode such assessments, simplifying both training and inference.

Our contributions are threefold:

- We formulate image-editing prompt refinement as an adaptive stop-or-refine policy that learns from evaluator feedback during training and directly decides whether and how to refine at inference, without explicitly generating an intermediate evaluation.
- We introduce two complementary roles of training-time feedback: *causal feedback* guides the construction of stop and refinement targets for offline learning, while *hindsight feedback* provides token-level supervision on policy-sampled outcomes for online self-distillation.
- Extensive experiments on ImgEdit-Bench and GEdit-Bench-EN show that FeedEdit outperforms strong state-of-the-art refinement methods, learns effective adaptive stop-or-refine behavior, and transfers across editors without retraining.

2 Preliminaries

2.1 Instruction-Guided Image Editing

Instruction-guided image editors such as FLUX.1 Kontext (Labs et al., 2025) and FLUX.2-klein-9B (Black Forest Labs, 2026) modify a source image according to a textual prompt. The prompt provides the interface through which user instructions control the editing behavior without modifying the editor itself. The goal is to satisfy the requested edit while preserving unrelated content.

2.2 Vision-Language Models

Multimodal large language models (MLLMs), particularly vision-language models (VLMs), extend autoregressive language models to condition on both visual and textual inputs. Given an image–text context, they generate textual outputs token by token, making them a natural interface for reasoning over editing results and producing revised editing prompts.

2.3 On-Policy Learning from Feedback

Learning from scalar rewards. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples a group of responses for the same input and evaluates them with scalar rewards. The rewards are normalized within the group to obtain relative advantages, which are used to update the policy toward higher-reward responses.

On-policy self-distillation. On-policy self-distillation trains a policy on its own sampled responses using a self-teacher derived from the same model. Self-Distillation Policy Optimization (SDPO) (Hübotter et al., 2026) conditions the self-teacher on environment feedback and successful responses sampled for the same input, such as compiler or runtime errors and successful solutions in coding tasks, and distills the teacher's next-token distribution into the policy using a reverse-KL objective.

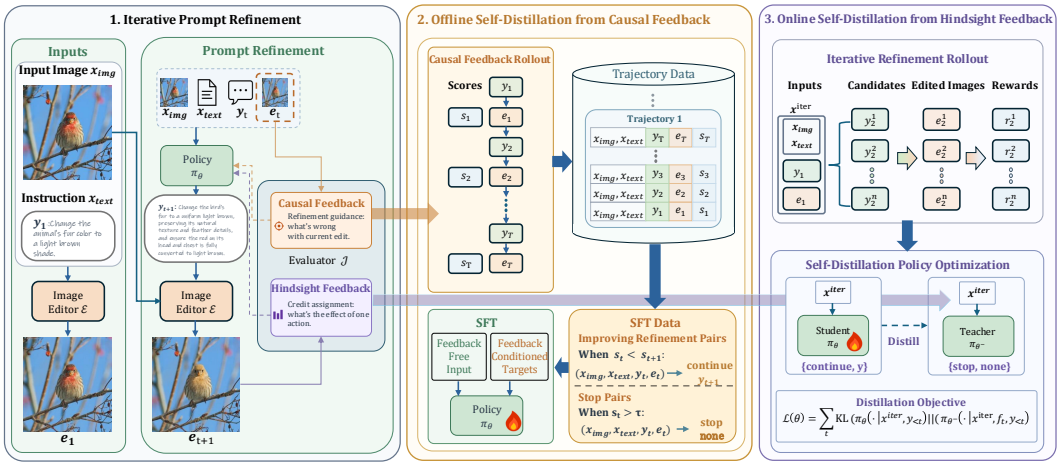

Figure 1: Overview of FeedEdit. **Left**: iterative prompt refinement, where each update is conditioned on the current editing result; feedback is used only during training and is absent from the deployed policy. **Middle**: causal feedback guides offline trajectory construction, from which stopping targets and refinement targets are extracted. **Right**: hindsight feedback from policy-sampled editing outcomes conditions a self-teacher for online self-distillation.

3 Methodology

FeedEdit learns an adaptive prompt-refinement policy (Section 3.1) for an editor using feedback available only during training (Figure 1). The outcome feedback supervises this policy in two complementary ways depending on when it is observed: feedback on the current edit provides *causal feedback* for constructing offline targets, whereas feedback on the outcome of a policy-sampled action provides *hindsight feedback* for online self-distillation (Sections 3.2–3.4).

3.1 Adaptive Prompt Refinement

Task formulation. Given a source image x_{img}, an editing instruction x_{text}, and an image editor $\mathcal{E}$, the goal is to produce a prompt y whose edited image $\mathcal{E}(x_{\text{img}}, y)$ fulfills the instruction while preserving unrelated content. Because refinement quality is determined by the resulting edit rather than the prompt itself, we define

$$Q(y;\, x_{\text{img}}, x_{\text{text}}) = \mathcal{J}\big(\mathcal{E}(x_{\text{img}}, y), x_{\text{img}}, x_{\text{text}}\big), \tag{1}$$

where $\mathcal{J}$ is an external evaluator used to assess the quality of editing results. **Iterative prompt refinement.** A fixed refinement schedule is undesirable because the current edit may already satisfy the instruction, while unnecessary additional edits can alter content that should be preserved. We therefore perform refinement adaptively over successive editor calls. Starting from the original instruction $y_1 = x_{\text{text}}$, the editor produces $e_t = \mathcal{E}(x_{\text{img}}, y_t)$ at iteration t. We define the refinement context as $z_t = (x_{\text{img}}, x_{\text{text}}, y_t, e_t)$, from which the policy predicts

$$(s_t, \tilde{y}_t) = \pi_\theta(z_t), \tag{2}$$

where $s_t = 1$ stops and returns the current edit e_t, while $s_t = 0$ updates the prompt as $y_{t+1} = \tilde{y}_t$. Each refined prompt is applied to the original image x_{img} rather than the preceding edited image. At inference, the policy operates directly on z_t without explicit assessment generation. Figure 2 illustrates targeted correction and early stopping under this procedure.

3.2 Feedback as Privileged Information

The evaluator feedback provides a useful signal of editing quality. Specifically, we leverage the instruction-compliance and content-preservation scores, together with textual feedback, returned by the evaluator $\mathcal{J}$. We distinguish two uses of this feedback according to when it becomes available relative to the refinement action.

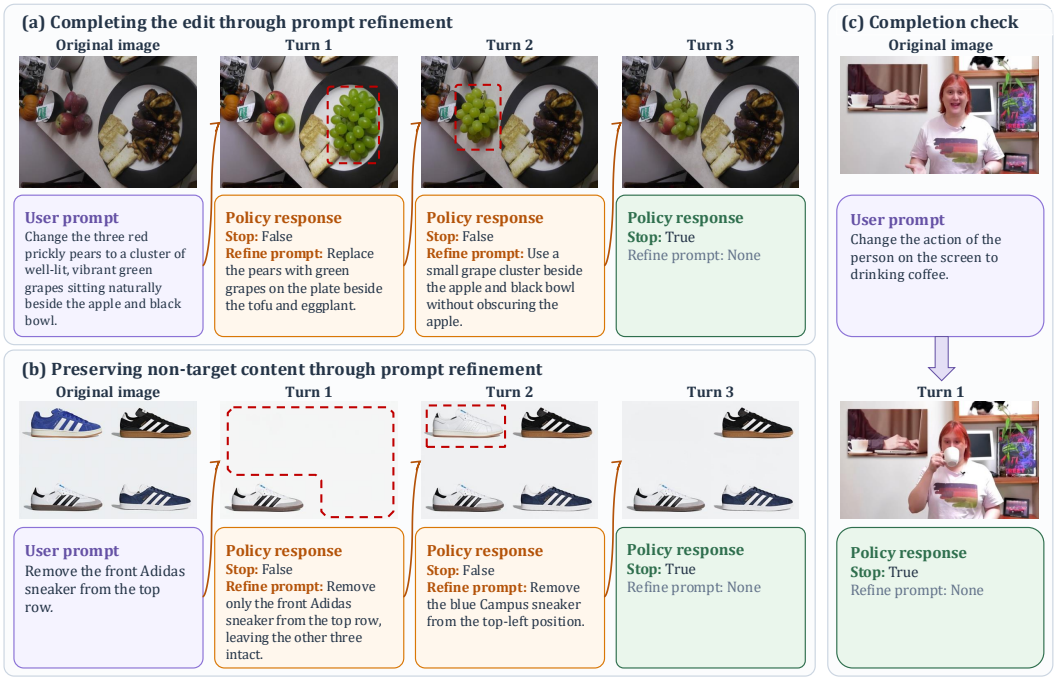

Figure 2: **Adaptive prompt refinement.** At each turn, FeedEdit either refines the prompt based on the current editing result or stops once the instruction is satisfied; each refined prompt is applied independently to the original image. Red dashed boxes highlight residual editing errors. **(a)** Refinement corrects the misplaced grapes and restores the unintentionally altered apple. **(b)** Refinement restores non-target sneakers that were mistakenly removed while preserving the requested shoe removal. **(c)** FeedEdit stops after the first edit because the instruction is already satisfied.

Causal feedback. Causal feedback evaluates the current edit e_t before the next action is generated. Its scores C_t and P_t indicate whether the instruction is satisfied and unrelated content is preserved, while the textual feedback f_t identifies the remaining discrepancy when further refinement is needed. Conditioning the base policy on f_t can produces a candidate corrective prompt. We use "causal" to denote feedback that guides the next action, rather than an estimate of a causal effect.

Hindsight feedback. Hindsight feedback is obtained after the current policy samples an action and the resulting image is evaluated. It therefore describes the outcome of a policy-generated action: the scores summarize the resulting edit quality, while the textual feedback identifies remaining errors. This outcome information provides supervision for refining actions sampled by the evolving policy.

Privileged self-distillation. We leverage self-distillation to learn from such evaluator signals. In both stages, the evaluator feedback is visible to the teacher but hidden from the deployed student model. In both stages, evaluator feedback is available to a teacher but hidden from the deployed student. For prompt generation, the teacher $\pi_{\mathcal{T}}$ conditions on the editing context z_t and evaluator feedback $\mathcal{F}_t$, whereas the student policy conditions only on z_t:

$$\pi_\theta(\tilde{y}_t \mid z_t) \xleftarrow{\text{refinement supervision}} \pi_{\mathcal{T}}(\tilde{y}_t \mid z_t, \mathcal{F}_t). \tag{3}$$

Causal feedback is used to construct offline refinement and stopping targets, whereas hindsight feedback provides privileged teacher context for online self-distillation. At deployment, the policy acts from z_t alone.

3.3 Offline Self-Distillation from Causal Feedback

We first train the stop-or-refine policy through offline self-distillation. The pretrained model uses causal feedback to construct candidate actions, while the student learns only from the feedback-free context z_t.

Feedback-guided trajectory construction. Let π_0 denote the pretrained model used to initialize the policy. At each trajectory step, the evaluator assesses the current edit e_t and returns instruction-compliance and content-preservation scores C_t and P_t, together with textual feedback f_t. Conditioned on the refinement context and causal feedback, the base model proposes the next prompt:

$$y_{t+1} \sim \pi_0(\cdot \mid z_t, f_t). \tag{4}$$

The candidate prompt is executed by the editor, and the resulting image is evaluated to continue the trajectory. From these trajectories, we construct two types of supervision, both using z_t alone as the student input. Trajectory-construction details are provided in Appendix F.1.

Stop pairs. To supervise stopping on satisfactory edits, we define $S_t = C_t + P_t$ and construct a stop pair when

$$S_t > \tau_{\text{stop}}, \qquad \tau_{\text{stop}} = 8. \tag{5}$$

The corresponding target is

$$(s_t, \tilde{y}_t) = (1, \text{null}),$$

teaching the policy to terminate refinement from the observed editing context.

Improving refinement pairs. To supervise corrective refinement, we retain a transition $(t, t+1)$ only if

$$C_{t+1} \geq C_t, \qquad P_{t+1} \geq P_t, \qquad S_{t+1} - S_t \geq \tau_{\text{re}},$$

where $\tau_{\text{re}} > 0$ is the minimum required improvement. Each retained transition provides the target

$$(s_t, \tilde{y}_t) = (0, y_{t+1}).$$

Thus, the policy learns to continue only from prompt updates whose improvement is verified by the resulting edit.

SFT objective. Let $a_t = (s_t, \tilde{y}_t)$ denote the target response. We optimize standard token-level cross-entropy over the full response:

$$\mathcal{L}_{\text{prompt}} = -\frac{1}{|a_t|} \sum_{i=1}^{|a_t|} \log \pi_\theta(a_{t,i} \mid z_t, a_{t,<i}). \tag{6}$$

The stopping decision and refined prompt are therefore learned jointly under the same next-token prediction objective.

3.4 Online Self-Distillation from Hindsight Feedback

Offline self-distillation provides a strong starting point by leveraging trajectories generated by the pretrained model. After offline self-distillation, we continue training on stop-or-refine actions sampled from the current policy. Executing these actions reveals their editing outcomes, from which hindsight feedback provides privileged supervision for a self-teacher, while the student continues to observe only z_t.

Hindsight feedback. Given a refinement context z_t, the policy samples a group of G responses $a_j = (s_j, \tilde{y}_j)$. A stopping response retains the current edit e_t, whereas a refinement response produces $\mathcal{E}(x_{\text{img}}, \tilde{y}_j)$. For each resulting image, the evaluator returns textual feedback f_j and instruction-compliance and content-preservation scores $C_j, P_j \in \{1, \ldots, 5\}$. The evaluator scores define the scalar outcome reward

$$r_j = \sqrt{C_j P_j}/5. \tag{7}$$

The same reward definition is used to filter contexts when constructing the online training dataset (Appendix F.2).

Hindsight feedback additionally determines whether a sampled refinement is successful. When supervising an unsuccessful response, if the same sampled group contains a successful refinement, we provide one such response as a successful refinement demonstration to the self-teacher. Thus, hindsight feedback supplies both outcome-specific correction signals and, when available, an example of a successful action under the same refinement context.

Feedback-conditioned self-teacher. For each response a_j, we construct a privileged teacher context $z_{t,j}^{\mathcal{T}}$ by augmenting z_t with its hindsight feedback (C_j, P_j, f_j) and, when available, a successful

refinement demonstration. The teacher $\pi_{\mathcal{T}}$ therefore conditions on privileged outcome information, whereas the student observes only z_t. Following the feedback-conditioned self-teacher formulation of Hübotter et al. (2026), for each response prefix $a_{j,<i}$ we use the token-level reverse-KL distillation loss

$$\ell_{j,i}^{\mathrm{SD}} = \mathrm{KL}\big(\pi_\theta(\cdot \mid z_t, a_{j,<i}) \,\big\|\, \pi_{\mathcal{T}}(\cdot \mid z_{t,j}^{\mathcal{T}}, a_{j,<i})\big)\,. \tag{8}$$

Thus, hindsight feedback distills outcome-informed supervision into prompt generation on policy-sampled actions.

4 Experiments

4.1 Experimental Setup

Models and training. We initialize the refinement policy π_θ from Qwen3-VL-8B-Instruct (Bai et al., 2025). For offline data construction, we use 297K tasks from Pico-Banana-400K (Qian et al., 2026) and OpenGPT-4o-Image (Chen et al., 2025), with Qwen3-VL-8B-Instruct as the refiner, FLUX.2-klein-9B (Black Forest Labs, 2026) as the editor, and GPT-5.6-Luna (OpenAI, 2026) with medium effort as the evaluator. After filtering and verification, we construct a balanced SFT set of 51,934 examples, comprising 25,967 refinement targets and 25,967 stop targets. For SFT, we fine-tune the initialized policy for two epochs. Online training starts from the SFT checkpoint using 800 contexts, with the same editor and evaluator as in offline construction. Evaluator feedback is used only during data construction and training-time rollout evaluation; the deployed policy receives no evaluator feedback at inference time. Appendices D and F provide details of the offline and online data construction procedures, the refiner and evaluator prompts, and the training hyperparameters.

Benchmarks and metrics. We evaluate instruction-guided image editing across diverse tasks on ImgEdit-Bench (Ye et al., 2025) and GEdit-Bench-EN (Liu et al., 2025). ImgEdit-Bench evaluates instruction adherence, image-editing quality, and detail preservation on a 1–5 scale. GEdit-Bench-EN reports semantic consistency (G_{SC}), perceptual quality (G_{PQ}), and their per-image geometric mean (G_{O}) on a 0–10 scale, with each metric macro-averaged over 11 editing categories. Appendix A provides the task composition, judging protocols, and aggregation rules.

Evaluation protocol. Given an original image and editing instruction, each method uses its own refiner to produce refined prompts, applies each refined prompt to the original image, and returns the image from the final editor call, with a budget of at most 5 editor calls. We further report Avg. Prompt Tok., the average token length of the refined prompt used to produce the final result under the Qwen3-VL-8B-Instruct tokenizer, and Eval. Gen., which indicates whether a method explicitly generates an intermediate analysis or evaluation before producing the refined prompt.

Baselines. *Pre-edit planning* includes DIM-4.6B-Edit (Zeng et al., 2026b), which uses structured editing blueprints to guide editing, and UnifiedThinker-7B (Zhou et al., 2026), which generates structured reasoning followed by an enhanced editing prompt. *Iterative refinement* includes ReasonEdit-S (Yin et al., 2026), which reflects on intermediate results to guide subsequent edits; EditThinker (Li et al., 2025), which generates critiques before refining prompts; and MIRA (Zeng et al., 2026a), which predicts atomic edits or stopping decisions from intermediate results. DIM and ReasonEdit-S are evaluated with their own trained editors, while the other refiners are paired with frozen external editors. Since MIRA is not publicly released, we report the results from the original paper. We also include no refinement and the pretrained Qwen3-VL-8B-Instruct refiner, which is used without further training with our refiner prompt template. FeedEdit keeps the editor frozen and directly predicts a refined prompt or a stop decision. We evaluate its causal-feedback SFT checkpoint and two online variants initialized from it: Vanilla GRPO with scalar rewards and our hindsight-feedback self-distillation, both trained on the same online data. Baseline inference configurations and our training details are provided in Appendix A.3 and Appendix D, respectively.

4.2 Main Results

As shown in Table 1, on FLUX.2-klein-9B, Qwen3-VL-8B-Instruct under our refiner prompt raises the scores from 4.28 to 4.35 on ImgEdit-Bench and from 7.90 to 7.97 on GEdit-Bench-EN. Causal-

Table 1: Final-turn editing quality and inference descriptors on ImgEdit-Bench and GEdit-Bench-EN. The causal-feedback SFT checkpoint already outperforms prior refinement baselines on both benchmarks, while hindsight-feedback training further improves both overall scores. Best and second-best scores are shown in bold and underlined, respectively.

Model	ImgEdit-Bench	GEdit-Bench-EN			Inference		
	Overall ↑ (/5)	Semantics ↑ (/10)	Quality ↑ (/10)	Overall ↑ (/10)	Avg. Prompt Tok. (I) ↓	Avg. Prompt Tok. (G) ↓	Eval. Gen.
Baselines							
DIM-4.6B-Edit (7B designer)	3.73	6.79	6.35	6.12	14.50	10.09	Yes
InstructPix2Pix editor							
InstructPix2Pix	2.29	–	–	–	–	–	No
+ MIRA	2.63	–	–	–	–	–	No
ReasonEdit-S native editor							
ReasonEdit-S (Thinking and reflection)	4.00	8.47	7.32	7.60	12.72	12.07	Yes
FLUX.2-klein-9B editor							
FLUX.2-klein-9B	4.28	8.65	<u>7.70</u>	7.90	14.43	9.84	No
+ EditThinker	4.50	8.84	7.68	8.07	46.52	54.99	Yes
+ Qwen3-VL-8B-Instruct	4.35	8.76	7.67	7.97	16.72	17.63	No
+ UnifiedThinker-7B	4.45	8.60	7.56	7.86	43.65	41.42	Yes
+ FeedEdit (causal only)	<u>4.51</u>	8.91	**7.71**	<u>8.09</u>	18.89	13.48	No
+ FeedEdit (+ GRPO)	<u>4.51</u>	<u>8.92</u>	7.69	8.05	18.72	13.79	No
+ FeedEdit (+ hindsight)	**4.54**	**8.99**	**7.71**	**8.13**	20.85	16.02	No

feedback SFT further improves the scores to 4.51 and 8.09, surpassing all other reported baselines on both benchmarks. Hindsight-feedback self-distillation brings additional modest but consistent gains, reaching 4.54 and 8.13. In contrast, Vanilla GRPO reaches 4.51 and 8.05, showing no consistent improvement over the SFT checkpoint and smaller gains than hindsight feedback. The hindsight variant also produces substantially more concise refined prompts than EditThinker for the returned image (20.85 vs. 46.52 tokens on ImgEdit-Bench and 16.02 vs. 54.99 on GEdit-Bench-EN), suggesting more targeted refinement. Appendix A.2 provides the category-level ImgEdit-Bench results.

Figure 3 illustrates how FeedEdit turns observed editing failures into targeted prompt refinements. For the background-change example, the original instruction broadly asks for a snowy landscape, which leads the editor to alter foreground content as well; FeedEdit narrows the refinement to applying snow to the environment while explicitly preserving the rabbit and cabbage leaves. For object extraction, it clarifies the original "extract the pineapple" request by removing surrounding content and presenting the pineapple as the sole subject. For compositional editing, it makes the two requested changes explicit by replacing the rice with a single hamburger while preserving the avatar's eating action. These examples show that FeedEdit can use the observed edit to localize what needs correction, resolve ambiguities in the original instruction, and express multiple editing constraints more precisely. More qualitative examples are provided in Appendix B.

Supervision efficiency. FeedEdit does not generate explicit evaluations at inference time and therefore does not require auxiliary evaluation tokens as supervised targets. With about 52K supervised examples, FeedEdit uses substantially less supervision than EditThinker's 140K examples (Li et al., 2025) while achieving higher overall scores on both ImgEdit-Bench and GEdit-Bench-EN. This yields a more supervision-efficient training formulation.

4.3 Ablation Studies

Effect of causal feedback on trajectory construction. We ablate causal feedback during trajectory construction by removing evaluator feedback from the refiner input while keeping all other settings fixed, using the same 1,000 source-image–instruction examples.

As shown in Table 2, causal feedback raises the final trajectory score $S = C + P$ from 7.233 to 7.746, where C and P denote instruction compliance and content preservation, respectively. It also produces more usable supervision, increasing stop pairs from 344 to 467 and improvement pairs from 911 to 1,072. Meanwhile, the repeated-prompt rate—the fraction of cases where the refiner chooses to continue but repeats the previous prompt—drops from 38.4% to 13.8%, suggesting that

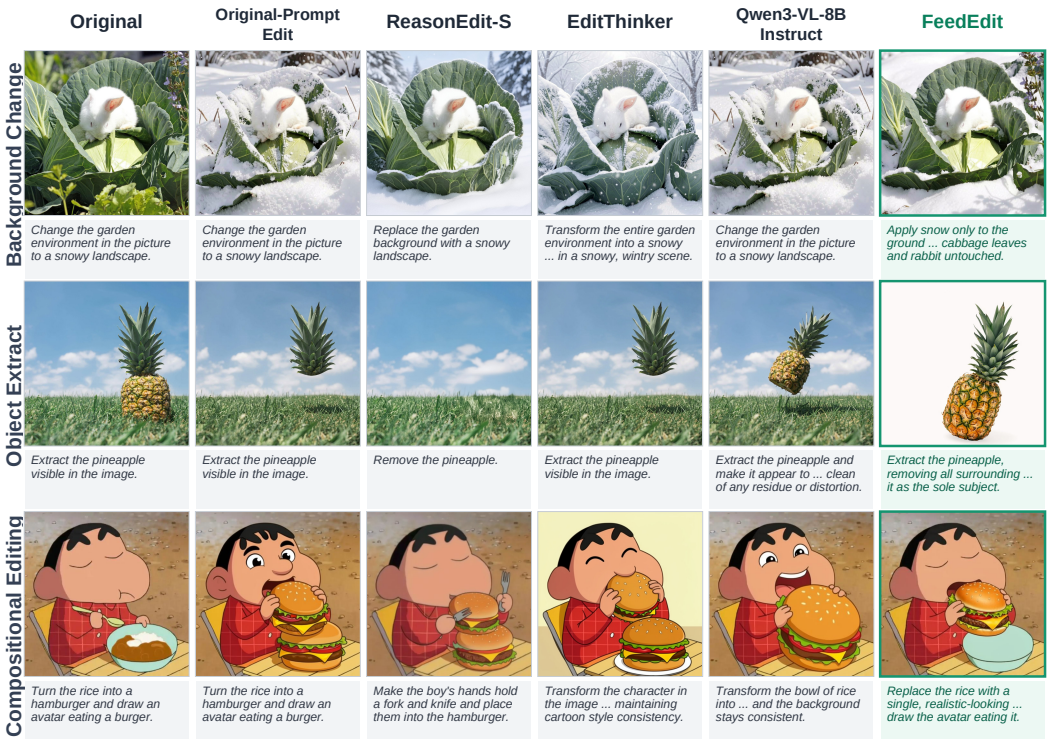

Figure 3: Qualitative comparison of prompt refinement across diverse image-editing tasks.

Table 2: Effect of causal feedback on offline trajectory construction using the same 1,000 source-image–instruction examples.

Construction variant	Final score $S \uparrow$	Stop pairs ↑	Improvement pairs ↑	Repeated-prompt rate ↓
With feedback	7.746	467	1,072	13.8%
Without feedback	7.233	344	911	38.4%

feedback helps it identify a concrete correction rather than stalling. Together, these results show that causal feedback improves both trajectory quality and supervision yield during offline construction. Additional details are provided in Appendix A.5.

4.4 Analysis

Adaptive stopping and refinement. We measure two complementary aspects of adaptive refinement: *completion discrimination*, defined as the turn-1 score of the policy-selected stop subset minus that of the continue subset, and *refinement gain*, defined as the improvement from the turn-1 score to the final score within the continue subset. As shown in Figure 4, FeedEdit achieves higher completion discrimination and refinement gain than Qwen3-VL-8B-Instruct and EditThinker on both benchmarks. This indicates that FeedEdit more effectively separates examples selected for stopping from those requiring further refinement, while also obtaining larger improvements on the latter. Appendix A.4 reports the stop- and continue-subset scores underlying these metrics.

Cross-editor transfer. Although trained with FLUX.2-klein-9B, FeedEdit transfers to FLUX.1-Kontext-dev (Labs et al., 2025) without retraining. As shown in Table 3, it continues to improve editing performance under the new editor, reaching 4.09 on ImgEdit-Bench and maintaining an advantage over EditThinker (4.08), UnifiedThinker-7B (3.98), and Qwen3-VL-8B-Instruct (3.91).

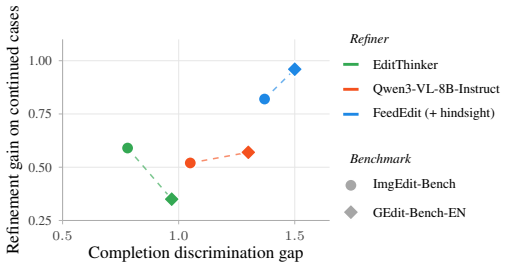

Figure 4: Completion discrimination and refinement gain for iterative refiners using FLUX.2-klein-9B.

Table 3: Cross-editor evaluation on ImgEdit-Bench with FLUX.1-Kontext-dev. FeedEdit is trained with FLUX.2-klein-9B and evaluated without retraining.

Model	Overall ↑ (/5)	Avg. Prompt Tok. ↓
FLUX.1-Kontext-dev	3.74	14.43
+ EditThinker	<u>4.08</u>	57.31
+ Qwen3-VL-8B	3.91	24.59
+ UnifiedThinker-7B	3.98	43.69
+ FeedEdit (+ hindsight)	**4.09**	24.78

5 Related Work

Pre-edit planning for image editing. Recent methods improve editing instructions by translating user intent into more structured prompts before execution. Draw-In-Mind (Zeng et al., 2026b) uses an external designer to construct editing blueprints, while Unified Thinker (Zhou et al., 2026) generates structured reasoning followed by an enhanced prompt. These approaches improve instruction specification before editing, but the refined prompt is determined without observing the resulting edit and therefore cannot adapt subsequent corrections to execution-specific errors. FeedEdit instead conditions each refinement on the observed editing result, allowing the policy to further correct the prompt or stop when the edit is already satisfactory.

Iterative refinement for image editing. Iterative methods use intermediate editing results to guide subsequent actions. EditThinker (Li et al., 2025) critiques edited images before refining prompts, while ReasonEdit-S (Yin et al., 2026) uses thinking and reflection to guide correction. MIRA (Zeng et al., 2026a) instead predicts atomic edits together with stopping decisions. FeedEdit differs by decoupling evaluation from action generation. Rather than generating an explicit assessment before each refinement, it uses evaluator feedback during training to learn prompt refinement from editing outcomes. This avoids both the dependency on generated assessments and the additional supervision and decoding they require. FeedEdit also keeps the image editor frozen, unlike Draw-In-Mind and ReasonEdit-S, which update the editor during training.

6 Conclusion

We presented **FeedEdit**, a feedback-driven framework for image-editing prompt refinement that treats editing feedback as privileged information during training. FeedEdit learns a direct stop-or-refine policy without requiring explicit evaluation generation: causal feedback constructs refinement and stopping supervision for offline learning, while hindsight feedback provides outcome-conditioned supervision for online self-distillation on policy-sampled actions. This design distills feedback-guided behavior into a policy that operates directly from the editing context at deployment. Experiments on ImgEdit-Bench and GEdit-Bench-EN show that FeedEdit achieves stronger editing performance than state-of-the-art refinement methods, learns effective adaptive stopping and refinement behavior, and transfers to a different image editor without retraining.

Limitations. Our approach remains bounded by the underlying editor's fine-grained manipulation capabilities. In particular, we observe failures when editing small text in dense layouts and when separating complex foreground objects from textured backgrounds. Prompt refinement can mitigate ambiguous instructions, but cannot fully overcome these low-level editing limitations. We provide representative failure cases in Appendix C.

AI USE STATEMENT

In this work, we used generative AI tools to produce LLM-based judgments for dataset construction and benchmark evaluation. We did not use generative AI tools to develop the research methodology, implement the core method, or interpret the main findings.

Additionally, we used generative AI tools for language editing to improve the readability of the manuscript, and for literature search and discovery to identify potentially relevant related work.

All AI-assisted outputs, including LLM-based judgments and edited text, were reviewed and verified by the authors. The authors take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work constructs a prompt-refinement dataset based on existing public data, with Qwen3-VL-8B-Instruct used for prompt refinement and GPT used as an evaluator. Given the scope and experimental setup of this study, we have not identified significant additional ethical risks beyond those associated with the source datasets.

REPRODUCIBILITY STATEMENT

Model choices and the experimental setup are specified in Section 4.1. Appendix D provides implementation details, including editor and refiner configurations, training hyperparameters, hardware, and random seeds, while Appendix D.1 documents the refiner and evaluator prompts and scoring criteria. Training-data sources and data-construction procedures are described in Appendices E, F.1, and F.2, together with Algorithms 1 and 2. Benchmark composition, judge models, metrics, aggregation rules, and baseline inference configurations are provided in Appendices A and A.3. We plan to publicly release the dataset, model checkpoints, and code.

REFERENCES

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.

Black Forest Labs. FLUX.2-klein-9B. `https://huggingface.co/black-forest-labs/FLUX.2-klein-9B`, 2026. Hugging Face model.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402. IEEE, 2023.

Zhihong Chen, Xuehai Bai, Yang Shi, Chaoyou Fu, Huanyu Zhang, Haotian Wang, Xiaoyan Sun, Zhang Zhang, Liang Wang, Yuanxing Zhang, et al. Opengpt-4o-image: A comprehensive dataset for advanced image generation and editing. *arXiv preprint arXiv:2509.24900*, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model

serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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*, 2025.

Hongyu Li, Manyuan Zhang, Dian Zheng, Ziyu Guo, Yimeng Jia, Kaituo Feng, Hao Yu, Yexin Liu, Yan Feng, Peng Pei, et al. Editthinker: Unlocking iterative reasoning for any image editor. *arXiv preprint arXiv:2512.05965*, 2025.

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*, 2025.

OpenAI. GPT-5.2, 2025. URL `https://developers.openai.com/api/docs/models/gpt-5.2`. Accessed: 2026-09-26.

OpenAI. GPT-5.6 Luna, 2026. URL `https://developers.openai.com/api/docs/models/gpt-5.6-luna`. Accessed: 2026-09-26.

Yusu Qian, Eli Bocek-Rivele, Liangchen Song, Jialing Tong, Yinfei Yang, Jiasen Lu, Wenze Hu, and Zhe Gan. Pico-banana-400k: A large-scale dataset for text-guided image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37226–37235, 2026.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Xiaoshuai Song, Yanan Wu, Weixun Wang, Jiaheng Liu, Wenbo Su, and Bo Zheng. Progco: Program helps self-correction of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 944–959, 2025.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Yuanzhu Peter Chen, and Tony Mak. Llms cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, 2024.

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, et al. Reasonedit: Towards reasoning-enhanced image editing models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23283–23293, 2026.

Ziyun Zeng, Hang Hua, and Jiebo Luo. Mira: Multimodal iterative reasoning agent for image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9563–9573, 2026a.

Ziyun Zeng, David Junhao Zhang, Wei Li, and Mike Zheng Shou. Draw-in-mind: rebalancing designer-painter roles in unified multimodal models benefits image editing. In *International Conference on Learning Representations*, volume 2026, pp. 84455–84487, 2026b.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. *Advances in neural information processing systems*, 36:31428–31449, 2023.

Sashuai Zhou, Qiang Zhou, Jijin Hu, Hanqing Yang, Yue Cao, Junpeng Ma, Yinchao Ma, Jun Song, Tiezheng Ge, Cheng Yu, et al. Unified thinker: A general reasoning modular core for image generation. *arXiv preprint arXiv:2601.03127*, 2026.

A Additional Experimental Results and Evaluation Details

This appendix collects the additional quantitative results and their associated evaluation details. Qualitative results are presented separately in Appendix B.

A.1 Benchmarks and aggregation

Evaluation is conducted on the Basic single-turn suite of ImgEdit-Bench (Ye et al., 2025) and the English subset of GEdit-Bench (Liu et al., 2025), which cover complementary edit types and quality criteria.

ImgEdit-Bench. The 737-example Basic suite contains one source image and one editing instruction per example and spans nine task types: action or motion change, object addition, attribute adjustment, background change, object extraction, hybrid editing, object removal, object replacement, and style transfer (Ye et al., 2025). The benchmark assesses instruction adherence, editing quality, and detail preservation. For image i, GPT-4o assigns a 1–5 score to each dimension; following the benchmark protocol, the editing-quality and detail-preservation scores are capped by the instruction-adherence score, and the per-image score is the arithmetic mean of the three dimensions. Category scores average the per-image scores within each task type, while the overall score averages them over the full evaluated set.

GEdit-Bench-EN. GEdit-Bench-EN contains 606 source-image–instruction pairs (Liu et al., 2025), grouped into 11 task types: background change, color alteration, material alteration, motion change, portrait editing, style change, object addition, object removal, object replacement, text editing, and tone transfer. GPT-4.1 assigns each edited image a 0–10 semantic-consistency score SC_i, which measures agreement with the editing instruction, and a 0–10 perceptual-quality score PQ_i, which measures visual quality and artifacts. The per-image overall score is their geometric mean, $O_i = \sqrt{\mathrm{SC}_i \mathrm{PQ}_i}$. For each $q_i \in \{\mathrm{SC}_i, \mathrm{PQ}_i, O_i\}$, the reported score is a task-balanced macro-average,

$$\overline{q}_{\mathrm{macro}} = \frac{1}{11} \sum_{g=1}^{11} \frac{1}{n_g} \sum_{i \in \mathcal{I}_g} q_i, \tag{9}$$

where $\mathcal{I}_g$ contains the n_g evaluated images in editing-task group g. Thus, G_SC, G_PQ, and G_O give equal weight to every task type, and G_O averages the per-image geometric means rather than taking the geometric mean of the aggregated G_SC and G_PQ values.

A.2 Full ImgEdit-Bench results

Table 4 reports the category-level ImgEdit-Bench results underlying the corresponding overall entries in Table 1.

A.3 Baseline Inference Configuration

Unless otherwise specified, we use the official default configuration for each baseline. For DIM, we use the variant with Qwen2.5-VL-7B-Instruct as the external prompt designer. For MIRA, we report the results from the original paper. UnifiedThinker, EditThinker, and FeedEdit use the same frozen editor within each shared-editor comparison.

A.4 Stopping-behavior analysis

Each row partitions a method's evaluated examples by its decision after the first edit and measures two complementary abilities. *Completion discrimination* is the turn-1 score of the stop subset minus that of the continue subset; a larger positive gap indicates that the policy more reliably stops on already satisfactory edits. *Refinement gain* is the final score minus the turn-1 score within the continue subset; a larger positive gain indicates more effective prompt refinement. Each policy determines its own stop and continue subsets. Scores and differences are computed from unrounded values and rounded independently for display.

Table 4: Category-level final-turn results on ImgEdit-Bench. Best and second-best scores are marked within the controlled FLUX.2-klein-9B block.

Model	ImgEdit-Bench Score ↑ (/5)										Inference	
	Action	Add	Adjust	Background	Extract	Hybrid	Remove	Replace	Style	Overall	Avg. Prompt Tok. ↓	Eval. Gen.?
Baselines												
DIM-4.6B-Edit (7B designer)	4.06	4.06	3.57	4.15	2.39	2.99	3.50	4.01	4.73	3.73	14.50	Yes
InstructPix2Pix editor												
InstructPix2Pix	1.81	2.19	2.87	2.00	3.01	–	1.66	2.10	2.19	2.29	–	No
+ MIRA	2.56	2.73	3.17	2.14	3.79	–	2.15	1.75	2.49	2.63	–	No
ReasonEdit-S native editor												
ReasonEdit-S (Thinking and reflection)	4.49	4.27	4.51	4.18	1.92	3.51	4.40	4.49	4.53	4.00	12.72	Yes
FLUX.2-klein-9B editor												
FLUX.2-klein-9B	4.83	4.60	4.60	**4.60**	2.42	**3.95**	4.59	4.58	4.88	4.28	14.43	No
+ EditThinker	**4.94**	4.64	4.58	4.39	3.73	3.89	**4.81**	**4.71**	4.81	4.50	46.52	Yes
+ Qwen3-VL-8B-Instruct	4.83	4.66	4.58	4.59	2.74	3.93	4.65	4.61	4.87	4.35	16.72	No
+ UnifiedThinker-7B	4.61	4.69	4.45	4.54	3.78	3.74	4.58	4.50	4.81	4.45	43.65	Yes
+ FeedEdit (causal only)	4.90	4.67	4.54	4.56	3.76	3.60	4.74	4.62	4.88	4.51	18.89	No
+ FeedEdit (+ hindsight)	4.91	**4.71**	**4.61**	4.54	**3.82**	3.82	4.76	4.60	**4.91**	**4.54**	20.85	No
FLUX.1-Kontext-dev editor												
+ UnifiedThinker-7B	4.50	4.15	3.91	4.15	3.43	3.35	3.43	4.28	4.47	3.98	43.69	Yes

Table 5: Stopping and refinement behavior on ImgEdit-Bench using policy-selected subsets and at most five editor calls.

Refiner	Score (/5)				
	Stop Turn 1	Continue Turn 1	Stop Turn 1 − Continue Turn 1	Continue Final	Continue Final − Turn 1
ReasonEdit-S (T+R)	4.07	3.95	+0.12	3.80	−0.15
FLUX.2-klein-9B					
EditThinker	4.59	3.81	+0.78	4.40	+0.59
Qwen3-VL-8B-Instruct	4.41	3.36	+1.05	3.88	+0.52
FeedEdit (causal only)	4.68	3.17	+1.51	4.07	+0.91
FeedEdit (+ hindsight)	4.72	3.35	+1.37	4.17	+0.82

ImgEdit-Bench. Table 5 reports GPT-4o overall scores on the nominal 1–5 scale.

GEdit-Bench-EN. Table 6 reports GPT-4.1 G_O scores on the 0–10 scale. Within each policy-selected stop or continue subset, scores are averaged within each represented editing-task group and then macro-averaged directly across those groups, giving every represented edit type equal weight.

A.5 Causal-feedback ablation during data construction

We randomly select 1,000 source-image–instruction tasks whose first edit remains below the acceptance threshold, $C + P \leq 8$, and therefore has room for refinement. Both variants use the same base Qwen3-VL-8B-Instruct refiner, FLUX.2-klein-9B editor, and GPT-5.6-Luna evaluator. The trajectory prompt and editor settings follow Appendices D.1 and D.2, and both variants use the same stopping rule and five-turn budget as the offline construction process in Appendix F.1.

The two variants share the exact first edit and use the same input structure at every continuation. In the feedback variant, the evaluator's textual feedback on the current edit is passed to the refiner when producing the next instruction; in the no-feedback variant, this field is left empty. The evaluator continues to score both variants, while all model configurations and stopping rules remain fixed, so the intervention isolates whether causal feedback is available to the refiner.

We report the final instruction-compliance score C, preservation score P, and their gains over the shared first edit. Improving pairs have non-decreasing component scores and a positive total-score gain. The stricter subset with $\Delta S \geq 2$, where $S = C + P$, provides a stronger improvement signal and is used for offline training. Starts with a pair measure how broadly each construction variant produces usable supervision.

Table 6: Stopping and refinement behavior on GEdit-Bench-EN using policy-selected subsets and at most five editor calls.

Refiner	Score (/10)				
	Stop Turn 1	Continue Turn 1	Stop Turn 1 − Continue Turn 1	Continue Final	Continue Final − Turn 1
ReasonEdit-S (T+R)	7.97	7.22	+0.75	6.80	−0.42
FLUX.2-klein-9B					
EditThinker	8.42	7.45	+0.97	7.80	+0.35
Qwen3-VL-8B-Instruct	8.17	6.87	+1.30	7.44	+0.57
FeedEdit (causal only)	8.21	6.65	+1.56	7.58	+0.92
FeedEdit (+ hindsight)	8.28	6.78	+1.50	7.74	+0.96

Table 7: Construction-time editing quality and training-pair yield over the 1,000 matched trajectory starts.

Metric	With feedback	Without feedback
Final compliance C	3.506	3.228
Final preservation P	4.240	4.005
Compliance gain	+0.635	+0.358
Preservation gain	+0.611	+0.376
Scored adjacent transitions	2,690	2,503
Improving pairs	1,072	911
Improving-pair rate	39.85%	36.40%
Starts with an improving pair	83.6%	73.6%
Improving pairs with $\Delta S \geq 2$	569	417
$\Delta S \geq 2$ pair rate	21.15%	16.66%
Starts with a $\Delta S \geq 2$ pair	51.2%	38.3%

Results. Causal feedback improves both components of editing quality: final compliance rises from 3.228 to 3.506 and final preservation from 4.005 to 4.240, with correspondingly larger gains over the shared first edit. It also produces more usable supervision, increasing the number of improving pairs from 911 to 1,072 and the number meeting the offline threshold $\Delta S \geq 2$ from 417 to 569. The fractions of tasks contributing these pairs rise from 73.6% to 83.6% and from 38.3% to 51.2%, respectively. Together with the lower repeated-prompt rate reported in Table 2, these results show that causal feedback helps the base refiner formulate effective corrections across a broader set of editing tasks.

B QUALITATIVE RESULTS

Figure 5 presents two types of qualitative results. We first compare the final edited images produced by different methods on examples from ImgEdit-Bench and GEdit-Bench-EN. We then provide additional FeedEdit editing examples across diverse edit types.

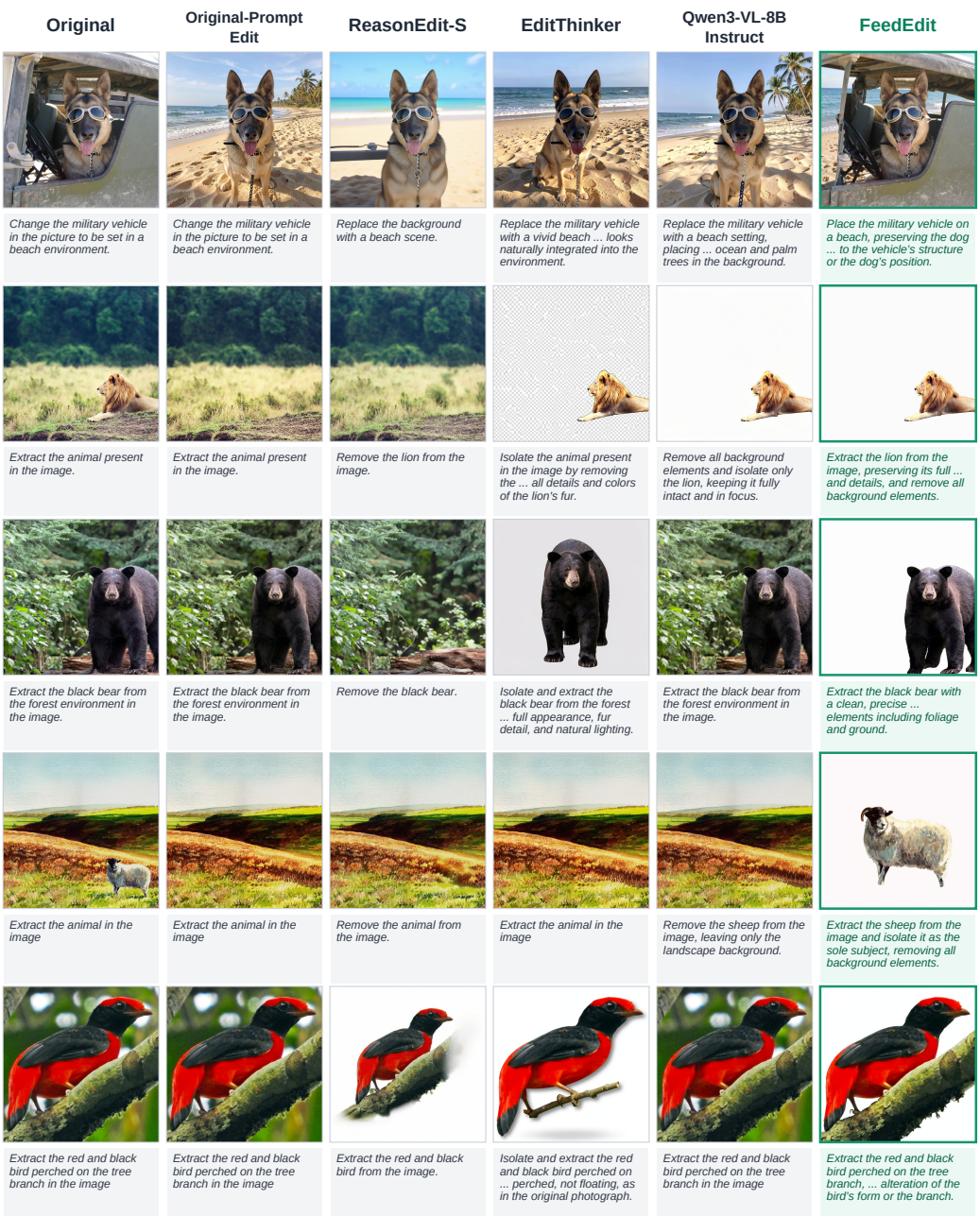

Figure 5: **Qualitative comparison with different prompt refining methods.**

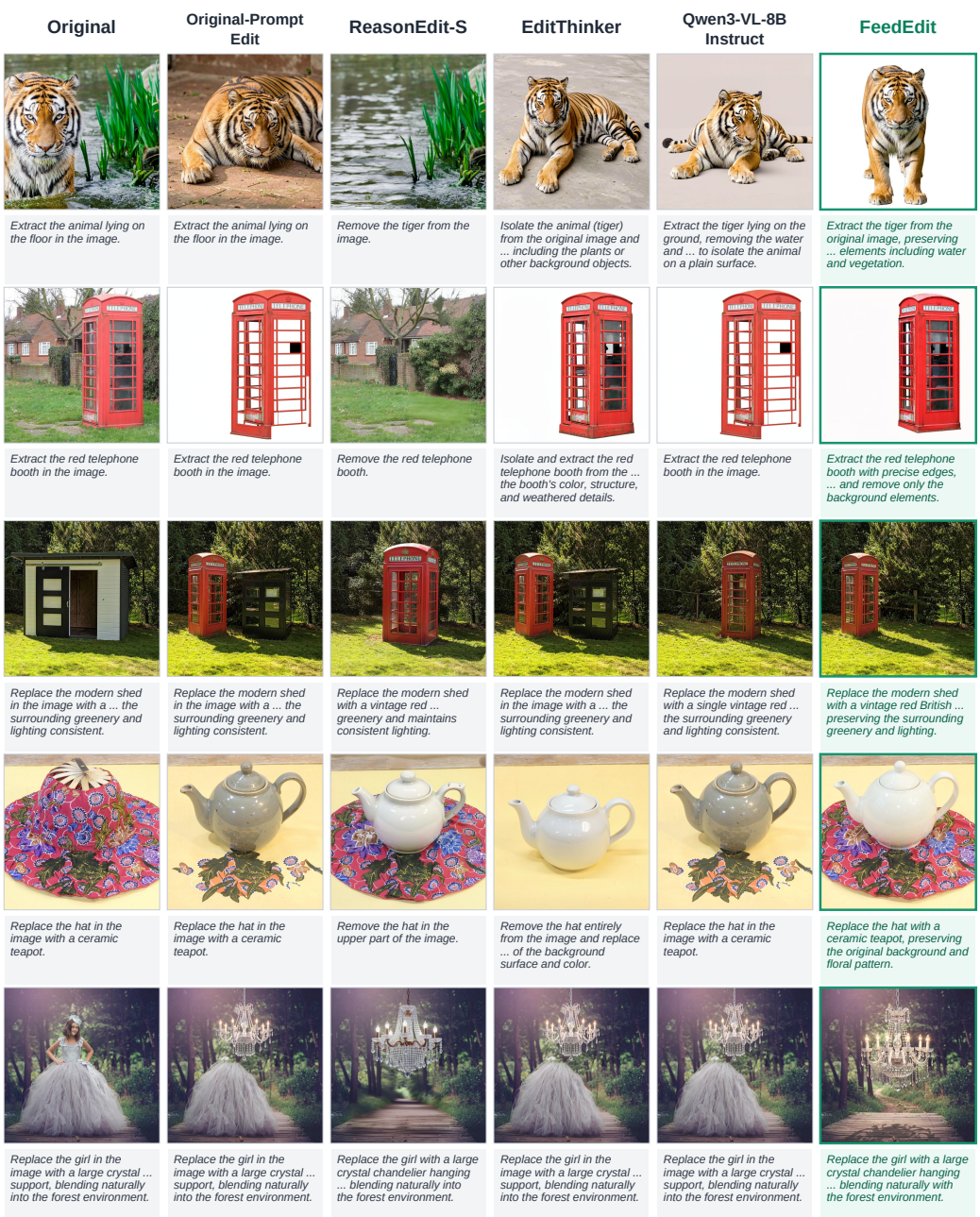

Figure 5: **Qualitative comparison with different prompt refining methods.**

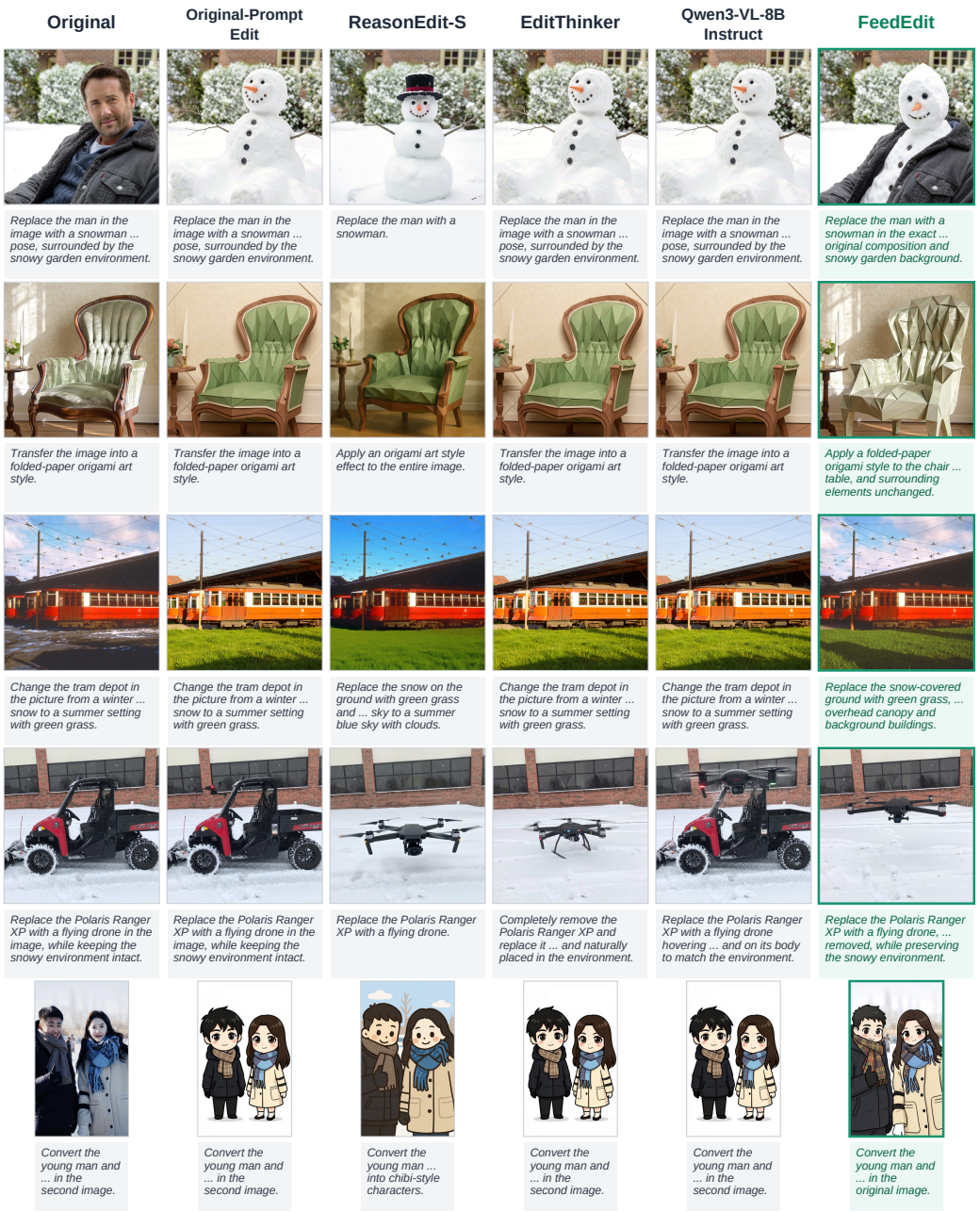

Figure 5: **Qualitative comparison with different prompt refining methods.**

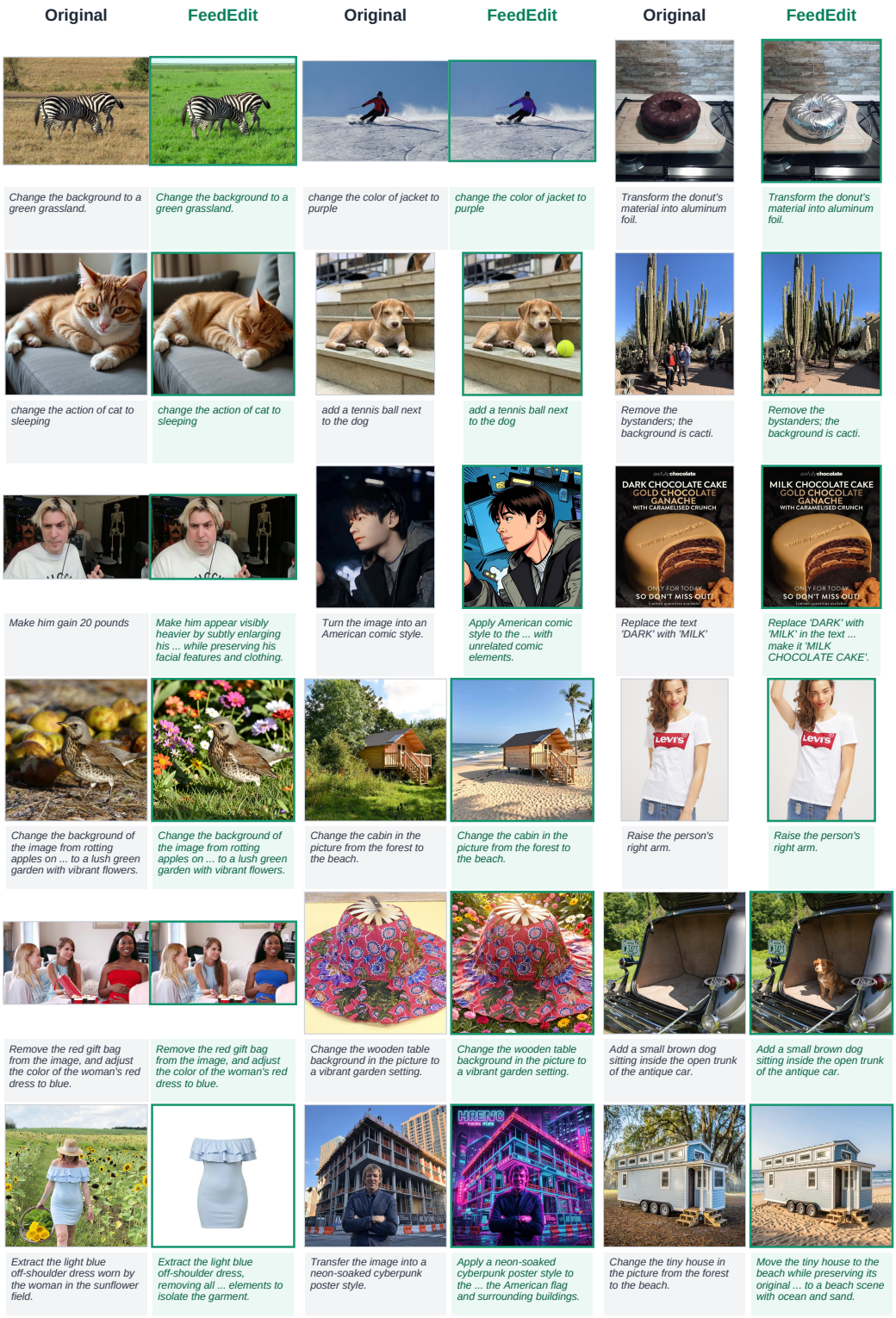

Figure 5: **Qualitative editing examples of FeedEdit.**

C FAILURE CASE STUDY

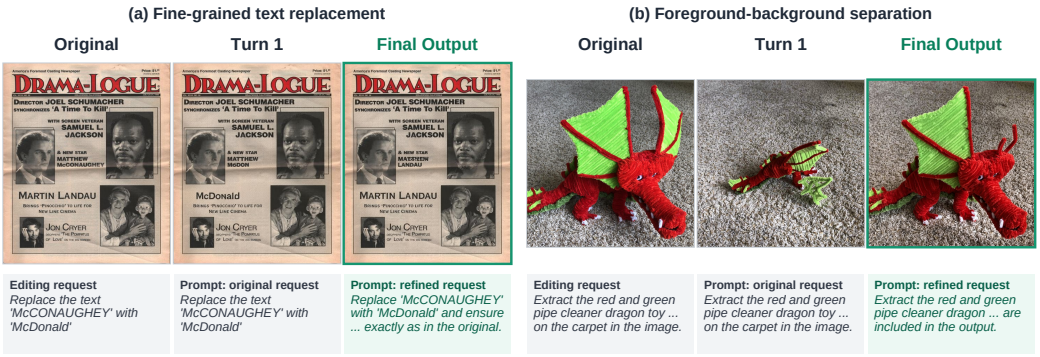

Figure 6: **Representative failure cases under editor capability limitations. (a) Fine-grained text replacement.** The editor fails to precisely localize and replace small text in a dense page layout while preserving surrounding content. **(b) Foreground–background separation.** When extracting a complex object from a textured background, the editor leaves background artifacts, indicating that prompt refinement remains constrained by the underlying editor's fine-grained separation capability.

D IMPLEMENTATION DETAILS

D.1 PROMPT TEMPLATES

We use separate templates for iterative refinement and evaluation. For iterative refinement, the offline trajectory-construction prompt includes the evaluator's causal feedback in `<eval_feedback_block>` and outputs only the refined instruction. The resulting targets are converted to the stop-aware JSON format described in Appendix F.1. During policy training and inference, the stop-aware template is used directly; `<eval_feedback_block>` is populated only for feedback-conditioned teacher contexts and left empty for the student and at inference. Both refinement templates require direct responses without chain-of-thought or auxiliary analysis.

The evaluator outputs instruction-compliance and content-preservation scores, causal feedback, a failure-mode label, and a brief reason. The scores and causal feedback are used for offline trajectory construction.

Iterative refinement prompt for trajectory construction (with causal feedback)

```
You are an expert in refining image-editing instructions.

Given:
1) Original image: the source image used for editing
2) Edited image: the result generated from the previous refined instruction
3) Original editing instruction: the user's intended editing goal
4) Previous refined instruction: the prompt used in the last edit

Comparative Analysis:
Compare the edited image with the original image and the original instruction. Identify:
- Missing or incomplete edits
- Incorrect or inaccurate edits
- Over-edited regions
- Unintended changes
- Elements that should have been preserved but were altered

Task:
Decide whether the edited image already satisfies the original instruction. If it does
not, refine the previously revised instruction for editing the original image, so that it
produces higher-quality edits.

Guidelines:
- Refine the instruction to address the identified issues
- If an intended edit is not properly reflected in the edited image, treat the
corresponding instruction as ineffective
- When an instruction is ineffective, strengthen or rewrite it instead of keeping it
unchanged
- Keep the instruction concise and avoid unnecessary detail

Output Requirement:
Return ONLY the refined instruction text.

<eval_feedback_block>

Original instruction:
<original_prompt>

Previous refined instruction:
<prev_refined_prompt>

The images are provided in order: first the Original Image, then the Edited Image.
<original_image>
<edited_image>
```

Adaptive refinement prompt for training and inference

```
You are an expert in refining image-editing instructions.

Given:
1) Original image: the source image used for editing
2) Edited image: the result generated from the previous refined instruction
3) Original editing instruction: the user's intended editing goal
4) Previous refined instruction: the prompt used in the last edit
```

```
Comparative Analysis:
Compare the edited image with the original image and the original instruction. Identify:
- Missing or incomplete edits
- Incorrect or inaccurate edits
- Over-edited regions
- Unintended changes
- Elements that should have been preserved but were altered

Task:
Decide whether the edited image already satisfies the original instruction. If it does
not, refine the previously revised instruction for editing the original image, so that it
produces higher-quality edits.

Guidelines:
- Refine the instruction to address the identified issues
- If an intended edit is not properly reflected in the edited image, treat the
corresponding instruction as ineffective
- When an instruction is ineffective, strengthen or rewrite it instead of keeping it
unchanged
- Keep the instruction concise and avoid unnecessary detail

Output Requirement:
Return ONLY a JSON object with exactly two fields:
{"stop": <true|false>, "refined_prompt": <string or null>}
Set "stop" to true when the edited image already satisfies the original instruction, and
set "refined_prompt" to null. Otherwise set "stop" to false and put the refined
instruction in "refined_prompt".

<eval_feedback_block>

Original instruction:
<original_prompt>

Previous refined instruction:
<prev_refined_prompt>

The images are provided in order: first the Original Image, then the Edited Image.
<original_image>
<edited_image>
```

Evaluator prompt

```
You are a professional image-editing evaluator.
Your task is to evaluate edited images by analyzing the following items in sequence:
1. Original Image: One reference image used for the edit operation
2. Edited Image: The final edited result, to be compared against the original
3. Editing Instruction: the prompt describing the intended edit

Comparative Analysis: Compare the edited result against the original image to assess:
• Whether the instruction was correctly understood and applied
• Whether any unintended changes were introduced outside the target region


Evaluation Criteria:

Edit Instruction Compliance (1-5)
Evaluate whether the requested edit is correctly and completely implemented.
5 - Fully compliant: All requested changes are clearly and correctly implemented; no
meaningful requested element is missing or incorrect.
4 - Mostly compliant: The main edit is correctly implemented, with only minor
inaccuracies, omissions, or visual imperfections.
3 - Partially compliant: The edit is recognizable, but one or more important requested
details are missing, incorrect, or ambiguous.
2 - Weakly compliant: Only a small portion of the requested edit is correctly implemented;
major requested changes are missing or incorrect.
1 - Non-compliant: The requested edit is not implemented, fundamentally incorrect, or
unrelated to the instruction.

Preservation (1-5)
Evaluate whether content outside the intended edit region remains unchanged.
5 - Excellent preservation: Non-target regions are effectively unchanged; the edit is
cleanly localized.
4 - Good preservation: Minor unintended changes are visible, but they do not materially
affect the image.
```

```
3 - Moderate preservation: Noticeable unintended changes occur outside the target area,
though the original content remains largely recognizable.
2 - Poor preservation: Significant unintended changes affect non-target content or overall
image consistency.
1 - Very poor preservation: Non-target regions are substantially altered, damaged, or
regenerated.


Feedback Instruction:
Identify the SINGLE most critical issue in the edited image.

Requirements:
- Focus on the most important problem
- State only what is wrong
- Do NOT restate correct parts
- If the requested edit was not achieved, describe what is actually visible and how it
differs from the requested result
- Keep it concise and specific.

Return only the following valid JSON object.
{
  "Edit Instruction Compliance": <integer 1-5>,
  "Preservation": <integer 1-5>,
  "Failure Mode": "<one of: Edit Instruction Compliance | Preservation | none | mixed>",
  "Brief Reason": "<short explanation>",
  "feedback": "<single most critical issue>"
}

Below are the images in order: first the Original Image, then the Edited Image.
<original_image>
<edited_image>

Editing instruction is:
<original prompt>
```

D.2 Editor Configuration

We keep the image editor frozen throughout data construction, policy training, and evaluation. FLUX.2-klein-9B (Black Forest Labs, 2026) is used for offline trajectory construction, online rollouts, and the main evaluation, with 4 sampling steps, guidance scale 1.0, and seed 0. Images are rendered at aspect-ratio-preserving resolutions with a maximum area of 1,048,576 pixels. All refined instructions are applied to the source image. Within each controlled evaluation, all refiners use the same editor checkpoint and generation settings. For cross-editor evaluation, we replace FLUX.2-klein-9B with FLUX.1-Kontext-dev (Labs et al., 2025) while keeping the refinement policy fixed.

D.3 Refiner Inference Configuration

FeedEdit uses Qwen3-VL-8B-Instruct (Bai et al., 2025) as the refinement-policy backbone and vLLM (Kwon et al., 2023) for inference. The policy takes the source image, current edited image, original instruction, and current editor prompt as input. The evaluator-feedback field is left empty at inference. We allow at most 5 editor calls per example, with the first call using the original instruction. Subsequent refinement prompts are applied to the source image.

D.4 Offline Self-Distillation Training Hyperparameters

We fine-tune Qwen3-VL-8B-Instruct while freezing the vision tower and multimodal projector. Training uses DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) with bfloat16 precision, a maximum image resolution of 1,048,576 pixels, a maximum sequence length of 4,096 tokens, and the `qwen3_vl_nothink` template.

We train for 2 epochs on 8 H800 GPUs with seed 42. We use a learning rate of 1×10^{-5} with cosine scheduling, a warmup ratio of 0.03, and weight decay of 1×10^{-6}. The per-device batch size is 4 with gradient accumulation 1, resulting in an effective batch size of 32. The final SFT checkpoint is used to initialize online training.

D.5 Online Self-Distillation Training Hyperparameters

Online training starts from the SFT checkpoint using 800 contexts randomly sampled from the 14,006 eligible trajectory tasks described in Appendix F.2. Each update uses 16 contexts with $N = 8$ responses per context, yielding 128 rollout samples. We train for one epoch, corresponding to 50 updates. The maximum prompt and response lengths are 8,192 and 256 tokens, respectively. Rollout sampling uses a temperature of 1.0 and top-$p = 1.0$, without top-k truncation.

For the feedback-conditioned teacher $\pi_{\mathcal{T}}$, we use an EMA teacher π_{θ^-} with coefficient 0.05, top-$K = 100$, and demonstration threshold $\delta_{\text{demo}} = 1.0$.

Both online variants use a learning rate of 5×10^{-6}, an optimization mini-batch size of 16, a micro-batch size of 1 per GPU, total 8 H800 GPUs. The + GRPO variant starts from the same causal-feedback checkpoint and uses the same online data and rollout configuration.

E Dataset Sources for Training Data Construction

Pico-Banana-400K (Qian et al., 2026) and OpenGPT-4o-Image (Chen et al., 2025) provide source images, natural-language editing instructions, and corresponding edited outputs.

Pico-Banana-400K. Pico-Banana-400K is a 400K-example text-guided image-editing dataset built from real images, covering object-level changes, scene composition, style transfer, text and symbol editing, human-centric edits, spatial and layout modifications, and photometric adjustments.

OpenGPT-4o-Image. OpenGPT-4o-Image is an 80K-example dataset for advanced image generation and editing. We use 40K editing examples spanning multi-operation composition, style control, text rendering, and fine-grained modifications.

F Offline and Online Self-Distillation Data Construction

F.1 Offline Self-Distillation Data Construction

Initial task pool and difficulty filtering. We assemble 257K single-turn editing tasks from Pico-Banana-400K (Qian et al., 2026) and 40K from OpenGPT-4o-Image (Chen et al., 2025), for a total of 297K source-image–instruction pairs. To identify tasks that can benefit from prompt refinement, we use FLUX.2-klein-9B to execute the original instruction on each source image. We then use GPT-5.2 (OpenAI, 2025) to evaluate the resulting edit along three dimensions—instruction compliance, content preservation, and visual quality—each on a 1–5 scale. We treat tasks scoring at least 4 on all three dimensions as already well solved and use the remaining, more challenging tasks for trajectory construction.

Causal-feedback trajectory construction. For these challenging tasks, we use Qwen3-VL-8B-Instruct as the instruction refiner, FLUX.2-klein-9B as the editor, and GPT-5.6-Luna (OpenAI, 2026) with medium reasoning effort as the trajectory evaluator. The two evaluators play consecutive roles: we use GPT-5.2 to identify initial edits with room for improvement and GPT-5.6-Luna to supervise their subsequent refinement trajectories.

Each trajectory starts from the original instruction $y_1 = x_{\text{text}}$, and all turns edit the same source image x_{img}. At turn t, the editor produces $e_t = \mathcal{E}(x_{\text{img}}, y_t)$. GPT-5.6-Luna evaluates instruction compliance C_t and content preservation P_t, rating each from 1 to 5. It also returns causal feedback f_t describing the most important remaining error. Conditioned on $(x_{\text{img}}, x_{\text{text}}, y_t, e_t, f_t)$, the refiner generates the next instruction y_{t+1}, which begins the next edit–evaluate–refine step. The trajectory terminates when $C_t + P_t > 8$, when the refiner repeats the current instruction, or after 5 editor calls. Since the scores are integer-valued, the quality threshold also ensures $C_t, P_t \geq 4$.

Algorithm 1 summarizes this process. Causal feedback acts as privileged information for the refiner during trajectory construction; the resulting training examples retain the editing context and verified action.

Algorithm 1 Trajectory generation with causal feedback

Input: source pair $(x_{\text{img}}, x_{\text{text}})$, editor $\mathcal{E}$, evaluator $\mathcal{J}$, refiner π_0, max turns T, stop threshold $\tau_{\text{stop}} = 8$
Output: trajectory τ
$y_1 \leftarrow x_{\text{text}}$; $\tau \leftarrow \emptyset$
for $t = 1$ to T **do**
  $e_t \leftarrow \mathcal{E}(x_{\text{img}}, y_t)$
  $(\mathbf{s}_t, f_t) \leftarrow \mathcal{J}(x_{\text{img}}, x_{\text{text}}, y_t, e_t)$
  Append $(y_t, e_t, \mathbf{s}_t, f_t)$ to τ
  if $\sum_k s_{t,k} > \tau_{\text{stop}}$ **then**
    break
  end if
  $y_{t+1} \leftarrow \pi_0(x_{\text{img}}, x_{\text{text}}, y_t, e_t, f_t)$
  if $y_{t+1} = y_t$ **then**
    break
  end if
end for

Algorithm 2 Offline training-pair extraction

Input: trajectories $\mathcal{T}$, refinement threshold τ_{re}, stop threshold $\tau_{\text{stop}} = 8$
Output: refinement pairs $\mathcal{S}_{\text{ref}}$, stop pairs $\mathcal{S}_{\text{stop}}$
$\mathcal{S}_{\text{ref}} \leftarrow \emptyset$; $\mathcal{S}_{\text{stop}} \leftarrow \emptyset$
for all $\tau = \{(y_t, e_t, \mathbf{s}_t)\}_{t=1}^{T} \in \mathcal{T}$ **do**
  $S_t \leftarrow \sum_k s_{t,k}$ for all turns t
  for $t = 1$ to $T - 1$ **do**
    if $\mathbf{s}_{t+1} \geq \mathbf{s}_t$, $S_{t+1} - S_t \geq \tau_{\text{re}}$, and $y_{t+1} \neq y_t$ **then**
      Add $((x_{\text{img}}, x_{\text{text}}, y_t, e_t), y_{t+1})$ to $\mathcal{S}_{\text{ref}}$
    end if
  end for
  for $t = 1$ to T **do**
    if $S_t > \tau_{\text{stop}}$ **then**
      Add $((x_{\text{img}}, x_{\text{text}}, y_t, e_t), \textsc{Stop})$ to $\mathcal{S}_{\text{stop}}$
      break
    end if
  end for
end for

From trajectories to offline training examples. Each trajectory provides two complementary forms of supervision: refinement pairs teach the policy how to improve an unsatisfactory edit, while stop pairs teach it to recognize a satisfactory result. For an adjacent transition $(t, t+1)$, we retain y_{t+1} as a refinement target when neither component score decreases, the total-score gain reaches the refinement threshold τ_{re}, and the instruction changes. A state whose total score exceeds the stopping threshold τ_{stop} becomes a stop pair. We set $\tau_{\text{re}} = 2$ and $\tau_{\text{stop}} = 8$, so refinement pairs improve the total score by at least 2 and stop pairs satisfy $C_t + P_t > 8$. Algorithm 2 summarizes both extraction rules.

For both pair types, the student input is the editing context $(x_{\text{img}}, x_{\text{text}}, y_t, e_t)$. The structured target is `{"stop": false, "refined_prompt": "`y_{t+1}`"}` for a refinement pair and `{"stop": true, "refined_prompt": null}` for a stop pair. Evaluator feedback guides target construction, while the student learns from the resulting context–action pair.

Mining the constructed trajectories produces 106,331 candidate records: 50,783 refinement candidates and 55,548 stop candidates. Applying the $\Delta S \geq 2$ improvement criterion retains 25,967 refinement pairs. We form a balanced dataset by selecting 25,967 stop pairs, comprising 18,182 states with $(C_t, P_t) = (5, 5)$ and 7,785 states with $C_t + P_t = 9$. The final offline dataset therefore contains 51,934 unique examples, evenly divided between refinement and stopping supervision.

F.2 Online Self-Distillation Data Construction

We construct the online self-distillation data from the trajectory records generated during offline data construction.

We first define each refinement state as the tuple (original image, original instruction, previous instruction). We remove states that qualify as offline stop or refinement examples, including those selected for offline training. From the remaining records, we retain tasks whose initial edit is generated by the original instruction and whose normalized reward satisfies $r \in [0.2, 0.7)$, where

$$r = \frac{\sqrt{C \cdot P}}{5}. \tag{10}$$

We further require each trajectory to contain at least two distinct validly scored instructions with unequal rewards. After state filtering and image validation, 14,006 tasks remain, from which we randomly sample 800 for online training. Different refinement states from the same source image or editing task may appear in both the offline and online stages.

For each selected context, the policy samples $N = 8$ responses. The corresponding prompts are executed on the source image using FLUX.2-klein-9B, and GPT-5.6-Luna (OpenAI, 2026) with medium reasoning effort evaluates instruction compliance C and content preservation P. We compute the reward for each response as

$$r = \frac{\sqrt{C \cdot P}}{5}. \tag{11}$$

Responses satisfying $r \geq \delta_{\text{demo}} = 1.0$ are used as demonstrations for feedback-conditioned teacher construction.