LEARNING TO PLAN AND ORCHESTRATE FOR OPEN-ENDED IMAGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern image editing models produce realistic results but struggle with abstract, multi-step instructions (e.g., "make this advertisement more vegetarian-friendly"). Prior agent-based methods decompose such tasks but rely on handcrafted pipelines or teacher imitation, limiting flexibility and decoupling learning from editing outcomes. We propose an *experiential framework for abstract, long-horizon image editing*: a planner generates structured atomic decompositions, an orchestrator selects tools and regions for each step, and a vision–language judge provides outcome-based rewards for instruction adherence, identity preservation and visual quality. The orchestrator is trained to maximize these rewards, and successful trajectories refine the planner. This tight coupling of planning and reward-driven execution yields more coherent, reliable edits than single-step or rule-based multi-step baselines. Our structured reward approximations further cut trajectory evaluation from exponential to linear in the number of subtasks, the learned policy transfers to unseen benchmarks (GEdit-Bench, MagicBrush), and gains hold even under a fully open-source reward model.

1 INTRODUCTION

Diffusion-based image editing now handles well-specified edits with high fidelity. Methods such as InstructPix2Pix (Brooks et al., 2023), Prompt-to-Prompt (Hertz et al., 2022), and larger editors like Flux Kontext (Labs et al., 2025) and Qwen-Image-Edit (Wu et al., 2025a) do well when an instruction maps to a single concrete change, such as *"add a hat to the man"* or *"change the car color to red."* However, real editing tasks are often not like this. For example, turning a student-focused loan ad into a campaign for rural audiences (Fig. 1) means reworking imagery, slogans, messaging, and setting together—well beyond a single atomic edit, and often calling for a different tool per subtask. Existing agent systems chain multiple steps but depend on handcrafted pipelines or teacher imitation (Yang et al., 2024; Zhenyu et al., 2024; Ji et al., 2025; Yeh et al., 2025), with execution order fixed in advance. Because the planner never sees its own outputs during training and tool choices are never checked against editing results, these systems often drift off-distribution and generalize poorly to open-ended instructions.

In this paper, we split the problem of open-ended, long-horizon image editing into *planning* and *orchestration*. The planner decomposes a high-level instruction into a checklist of atomic subtasks and trains on plans it samples itself, keeping training aligned with inference. The orchestrator then selects tools and image regions, runs the edits, and is scored by a Vision-Language Model (VLM) judge on instruction adherence, identity preservation, and visual quality. These rewards supervise tool selection directly, tying each choice to how the edit actually turns out. A final refinement stage prunes subtasks that cannot be executed. A key challenge is that training this system is harder than standard supervision. No dataset of abstract multi-step plans exists, tool selection is ambiguous and context-dependent, and many different outputs can satisfy the same instruction. Editing tools are also costly to run, which makes exploring full trajectories intractable. We therefore approximate a plan's reward as the sum of independently judged subtask rewards. This lets us precompute scores over tool–region pairs and reduces the cost of scoring a plan from $O(B^L)$ to $O(LB)$ (Sec. B.5), where L is the number of subtasks and B the number of tool–region candidates per subtask. Across our experiments, the framework produces more reliable and instruction-faithful edits than both single-step generators and multi-step agent baselines.

Contributions. We cast abstract, open-ended editing as a joint planning-and-orchestration problem, enabling multi-step reasoning beyond single-step generation. Within this framework, the planner learns multi-step decompositions from its own checklist-guided samples, reducing distribution shift, while an experiential orchestrator selects tools and regions through a reward-driven policy grounded

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.

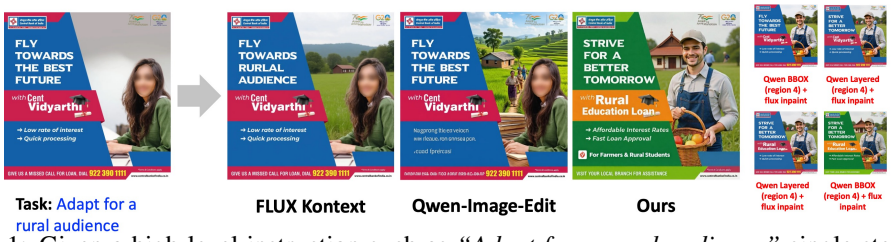

Figure 1: Given a high-level instruction such as *"Adapt for a rural audience,"* single-step editors (e.g., Flux Kontext and Qwen-Image-Edit) struggle to adapt visual themes, text, and audience context at once while preserving the original layout and identity. Our framework instead splits the task into structured subtasks and orchestrates multiple tools from outcome-based feedback. The four images right of our result show intermediate edits from automatically selected tool–region pairs, corresponding to the subtasks: (1) *replace the background with a rural scene featuring a farmer holding a basket of produce, keeping existing text intact*; (2) *replace 'FLY TOWARDS THE BEST FUTURE' with 'STRIVE FOR A BETTER TOMORROW'*; (3) *replace 'Cent Vidhyarthi' with 'Rural Education Loan'*; and (4) *add a village house or temple to reinforce the rural setting*).

in judged edits rather than handcrafted rules. Finally, we prune infeasible subtasks using orchestration feedback and reach state-of-the-art results on open-ended editing, verified on multiple unseen benchmarks and paired with an analysis of computational cost and judge reliability.

2 Related Work

Controllable image editing with diffusion models. Diffusion models edit images well from text (Rombach et al., 2022; Podell et al., 2023). Training-free methods like SDEdit and Prompt-to-Prompt (Meng et al., 2021; Hertz et al., 2022; Parmar et al., 2023; Cao et al., 2023; Hertz et al., 2024) steer the denoising process but stay local and tend to over-edit or miss the instruction. Training-based methods such as InstructPix2Pix and MagicBrush (Brooks et al., 2023; Zhang et al., 2024a) add robustness from paired supervision, and later work adds masks, boxes, or drag inputs for spatial precision (Li et al., 2023; Wang et al., 2024a; Mou et al., 2023; Shi et al., 2024; Nie et al., 2024). All assume well-specified, low-level instructions and often need manual controls; we instead target abstract, open-ended instructions requiring multi-step reasoning and coordinated tool use.

Multimodal LLMs for image editing and planning. One line of work has LLMs write code to call specialized modules, splitting a task into tool-executable subproblems (Gupta & Kembhavi, 2023; Surís et al., 2023; Hu et al., 2024; Huang et al., 2026). Multimodal LLMs (MLLM) (Liu et al., 2023; Zhu et al., 2023; Liu et al., 2024) bring this to editing: MGIE (Fu et al., 2023) rewrites instructions for a diffusion editor, and VLM agents break complex requests into steps for a fixed editor (Yang et al., 2024; Zhenyu et al., 2024; Ji et al., 2025; Yeh et al., 2025). Closest to our multi-tool setting, pipelines like GenArtist (Wang et al., 2024b) and X-Planner (Yeh et al., 2025) route between editors with fixed or prompt-based rules; we compare against representative multi-tool systems and consistently outperform them (Sec. 4.3). All are training-free or imitate teacher plans, so they never learn from real edits—the planner is not trained on its own plans and tool selection is not policy-optimized. Ours couples checklist-guided planning with experiential orchestration, learning tool and region selection from judged edits.

Experiential learning for long-horizon reasoning. RL has strengthened long-horizon reasoning in language models via step decomposition and iterative refinement (OpenAI, 2024; Guo et al., 2025; Wei et al., 2022), and multimodal settings via visually grounded chain-of-thought (Liu et al., 2025b; Huang et al., 2025). These refine a single model end to end; we instead learn a policy over multiple editing tools and regions to maximize a learned judge's reward. As diffusion editors are expensive, online RL over full trajectories is impractical, so we use structured reward approximations that keep optimization tractable while preserving credit assignment.

3 Approach

We propose an *experiential learning framework for long-horizon, open-ended image editing*. Abstract edits demand both high-level reasoning and low-level tool execution, which we learn through interaction with editing tools and feedback from a learned judge.

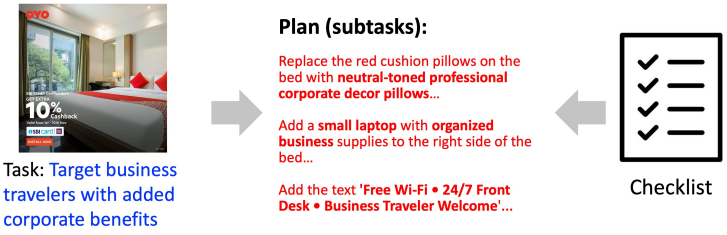

Figure 2: **Checklist-guided planner (Stage 1).** Given an advertisement and a high-level instruction (left, e.g., "Target business travelers with added corporate benefits"), the planner produces an ordered sequence of subtasks (center) addressing the checklist items (right)—here, adapting the room aesthetics for a professional audience, adding work-related objects, and introducing business-oriented text. Checklists, authored by human or LLM annotators, enforce coverage and interpretability while letting the planner produce modular plans for downstream orchestration.

Given an input image x and instruction I, our goal is an edited image $\hat{x}$ that fulfills I while preserving visual quality and essential details of the original. We split this into a **Planner**, which generates a structured sequence of sub-tasks, and an **Orchestrator**, which selects tools and regions to execute each step. Training is guided by an MLLM-based judge scoring correctness, visual quality, and consistency with the input. This decomposition addresses two problems at once: abstract instructions need multi-step, heterogeneous operations, and end-to-end optimization over full trajectories is too expensive—so we pair structured decomposition with efficient reward approximation.

3.1 Stage 1: Planner via Checklist-Guided Self-Training

Given an image x and high-level instruction I, the planner (a multimodal LLM) generates an ordered sequence of sub-tasks $\mathcal{P} = \{s_1, \ldots, s_T\}$, each a structured editing step (e.g., "add a laptop and organized business supplies to the bedside table"). It emits one subtask at a time, conditioned on (x, I) and all previous subtasks (Eq. 1), so ordering and dependencies come from the model rather than a fixed template. This turns an abstract objective into executable atomic operations, enabling modular, interpretable multi-step editing.

Checklist Construction. To stay close to the planner's native distribution (on-policy), we let it generate its own plans, rather than imitate a teacher model as was done in (Yeh et al., 2025). To nevertheless provide supervision over what a valid plan should satisfy, we introduce a *checklist* $\mathcal{C} = \{c_1, \ldots, c_K\}$: a short set of natural-language criteria (e.g., product substitution, semantic alignment, layout coherence). The planner is prompted with $(x, I, \mathcal{C})$ to generate plans satisfying every item (Fig. 2). Unlike prior checklist-based reward alignment for LLMs (Viswanathan et al., 2025), we use checklists to guide structured plan generation in long-horizon editing.

Checklists are not hand-tuned per image: annotators or an LLM can author them once from the instruction, and the same criteria templates carry across advertisement types, so a new domain only needs its relevant criteria specified. They are used *only* during data construction to encourage coverage—at inference the fine-tuned planner produces subtasks directly from (x, I) without any checklist (see *Self-Supervised Fine-Tuning* below).

This serves two ends: it enforces coverage, so the planner addresses all relevant aspects rather than producing partial plans, and it gives modular, human-interpretable supervision without gold-standard plans, avoiding the brittleness of hard-coded templates. Appendix B.2 shows that checklist-guided plans give greater coverage and more contextual edits than plans generated without one.

Self-Supervised Fine-Tuning. Let $\mathcal{P}^* = \{s_1^*, \ldots, s_T^*\}$ be the checklist-guided plan, where each sub-task $s_t^* = (s_{t,1}^*, \ldots, s_{t,N_t}^*)$ is a token sequence and each list element is a distinct operation. We fine-tune the planner to reproduce the full plan from (x, I) alone via autoregressive likelihood maximization:

$$\mathcal{L}_{\text{planner}} = \mathbb{E}_{(x,I)\sim\mathcal{D}}\left[-\sum_{t=1}^{T}\sum_{j=1}^{N_t} \log p_\theta\big(s_{t,j}^* \mid x, I, s_{<t}^*, s_{t,<j}^*\big)\right], \tag{1}$$

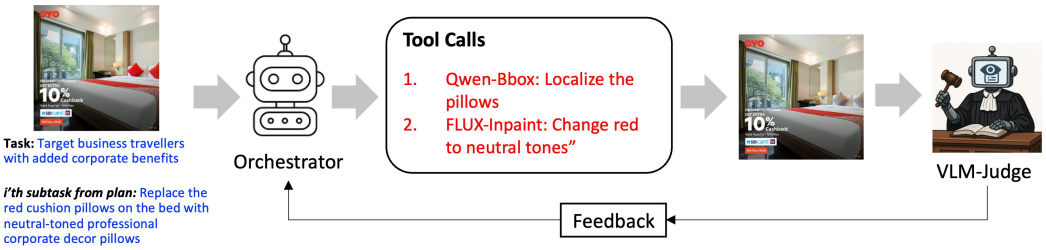

Figure 3: **Reward-driven orchestration (Stage 2).** Given an input image and a high-level instruction (e.g., "Target business travelers with added corporate benefits"), the orchestrator picks tools and regions to execute a subtask (e.g., replacing the red pillows with neutral professional decor). A VLM judge then scores the result on instruction adherence, identity preservation, and visual quality, and this feedback improves future tool/region selection.

where $\mathcal{D}$ is the training distribution, $s^*_{<t}$ the tokens of preceding sub-tasks, and $s^*_{t,<j}$ the tokens before position j within sub-task t.

Modeling the full plan autoregressively captures cross-subtask dependencies essential for long-horizon editing (e.g., a slogan change may depend on a prior object substitution) and keeps sequencing globally consistent. Since supervision comes from the planner's own checklist-prompted samples, training is a self-distillation that stays near the model's native distribution rather than external demonstrations, reducing inference-time distribution shift and improving robustness over pure off-policy imitation (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026). At inference the checklist is dropped and the planner generates a structured multi-step plan directly from (x, I).

3.2 Stage 2: Orchestrator via Reward-Driven Tool Selection

Given $(x, I, \mathcal{P})$, the orchestrator (a multimodal LLM with parameters ϕ) selects a tool a_t and image region r_t for each sub-task s_t. Tools (Sec. 3.5) are token sequences describing editing operations (e.g., object replacement, style transfer, text editing); regions are either the full image or candidate object/text areas from segmentation or bounding-box models. Casting tool and region selection as language generation integrates it directly into the LLM with no task-specific control logic, and keeps the action space open-ended—new tools are added simply by appending their descriptions.

Executing the sequence yields the edited image:

$$\hat{x} = f_{a_T, r_T} \circ \cdots \circ f_{a_1, r_1}(x), \tag{2}$$

where f_{a_t, r_t} applies tool a_t to region r_t when applicable. Sequential composition lets later edits build on earlier ones, which is essential for long-horizon tasks.

Reward Function. A strong MLLM-based judge (OpenAI, 2026) assigns a scalar reward $R(\hat{x}, x, I)$ from the edited image $\hat{x}$, the original x, and the instruction I, scoring instruction adherence, identity preservation, and visual quality (layout fidelity, realism; see Fig. 3). Since many outputs can satisfy the same instruction, a scalar reward gives flexible supervision without pixel-level alignment, and provides only outcome signals rather than dense token-level supervision. The judge is chosen through qualitative analysis and is *not* fine-tuned on our data, so it does not overfit to task-specific idiosyncrasies. Details are in Appendix C.2. Our user studies (Sec. 4.1) show these gains transfer to human preference, so the policy is not merely overfitting the judge. Sec. 4.2 gives a dedicated reliability and robustness analysis of the reward, including a fully open-source reward-model variant.

Policy Optimization Objective. We aim to maximize the reward of the full editing trajectory. Given tool–region decisions $(a_{1:T}, r_{1:T})$, executing the edits produces $\hat{x}$, scored by the judge as $R(\hat{x}, x, I)$, so we optimize the expected trajectory reward:

$$\max_{\phi} \mathbb{E}_{(a_{1:T}, r_{1:T}) \sim \pi_\phi} \big[R(\hat{x}, x, I) \big]. \tag{3}$$

Optimizing at the trajectory level encourages coordinated decisions, since later edits depend on earlier tool and region choices. In practice we sample candidate trajectories, keep the high-reward ones $(a^*_{1:T}, r^*_{1:T}) = \arg\max_{(a_{1:T}, r_{1:T})} R(\hat{x}, x, I)$ (using all when several score comparably), and

train the orchestrator to reproduce them by maximizing their likelihood:

$$\mathcal{L}_{\text{orch}} = -\mathbb{E}_{(x,I,\mathcal{P},a^*_{1:T},r^*_{1:T})}\left[\sum_{t=1}^{T}\log \pi_\phi\big(a^*_t, r^*_t \mid x, I, \mathcal{P}, a^*_{<t}, r^*_{<t}\big)\right]. \tag{4}$$

This aligns training with inference, grounding tool and region selection in empirically successful trajectories while staying tractable.

Efficient Reward Approximation. Learning high-reward actions requires exploring tool and region choices, but scoring a full trajectory is costly due to sequential diffusion calls, and enumerating all candidate sequences offline is infeasible since tool–region combinations grow exponentially with the number of sub-tasks. For a plan of L subtasks with B candidates each, exhaustively scoring full trajectories costs $O(B^L)$ generations. The two approximations below exploit the compositional structure of high-level edits to score each candidate independently per subtask, cutting this to $O(LB)$ and letting all tool–region rewards be precomputed once (Sec. B.5).

Additive Reward Approximation. Many edits are semantically distinct, largely independent operations (object replacement, slogan modification, background recoloring), and a high-quality result requires each to succeed. We therefore approximate the trajectory reward as a sum of sub-task contributions, $R(\hat{x}, x, I) \approx \sum_{t=1}^{T} R_t$, where R_t reflects whether sub-task s_t was completed. This is a *tractable approximation*, not an exact decomposition, but it fits our setting: our edits are mostly background or object/text operations on distinct regions, and checklist-guided plans tend to target distinct regions across subtasks, yielding mostly independent sequences. Strongly global operations (full palette shifts, complete background replacement) can break additivity by disrupting earlier edits, but these are rare for modern editors (FLUX, Qwen-Image-Edit) under the region-localized subtasks our planner produces. We quantify the residual coupling in Sec. 4.6.

Original-Image Independence Approximation. Since these operations are largely independent, a tool's effect usually depends only weakly on prior edits (e.g., product replacement rarely depends on an earlier background change). We therefore estimate a tool's contribution by evaluating it on the original image rather than intermediate edits: with x_{t-1} the image before applying (a_t, r_t), we approximate $R_t\big(f_{a_t,r_t}(x_{t-1}),\, x,\, I\big) \approx R_t\big(f_{a_t,r_t}(x),\, x,\, I\big)$. This is what enables offline precomputation—decoupling a tool's reward from the intermediate image lets tool–region rewards be evaluated once on x and reused across trajectories. It is an approximation rather than an identity; we validate it and quantify the small fraction of failing subtasks in Sec. 4.6.

Together, these let us precompute all tool–region candidates and rewards $\{(a, r, R_{a,r})\}$. For each sub-task we take the highest-reward tools and train the orchestrator to predict them.

3.3 Closing the Loop: Plan Refinement

To align planner and orchestrator, we prune sub-tasks whose maximum achievable reward falls below a threshold, $\max_{a,r} R_{a,r}(s_t) < \tau$; these correspond to operations the toolset cannot support. We then retrain the planner on the pruned plans so it reflects the feasible action space, and train the orchestrator only on subtasks scoring above τ. This grounds high-level reasoning in executable actions, enabling robust long-horizon editing without handcrafted pipelines.

The judge thus plays two roles. For *orchestrator supervision*, it supplies the outcome rewards $R_{a,r}(s_t)$ that train the tool/region policy (Sec. 3.2). For the *planner constraint*, its maximum achievable reward per subtask, $\max_{a,r} R_{a,r}(s_t)$, measures executability, and subtasks below τ are pruned before the planner is retrained. The same signal therefore optimizes the orchestrator and steers the planner toward feasible decompositions, closing the loop between reasoning and execution.

3.4 Inference via Verifier-Guided Selection

To improve robustness during sequential editing, we add a lightweight verifier that scores intermediate edits. Given the original image x, a sub-task s_t, and the edit $\tilde{x}_t = f_{a_t,r_t}(x_{t-1})$, it predicts a score for sub-task correctness, identity preservation, and visual quality. We distill teacher scores from the same VLM judge used in training (OpenAI, 2026) into a smaller VLM (Bai et al., 2025) for efficiency. For each sub-task, the orchestrator proposes a distribution over tool–region pairs; we take the top-k by policy likelihood, execute them, and re-rank with the verifier,

$(a_t^*, r_t^*) = \arg\max_{i \in \{1,\dots,k\}} \text{Verifier}\big(f_{a_t^{(i)}, r_t^{(i)}}(x_{t-1}), x, s_t\big)$, carrying the best edit to the next step. This proposal–re-ranking reduces error accumulation while staying tractable ($k = 3$ or 5 works well). Since the verifier scores edits on the true intermediate image x_{t-1}—unlike the training-time original-image approximation—it also guards against the residual coupling of Sec. 4.6. After all sub-tasks, a lightweight refinement improves coherence while preserving the intended edits.

3.5 TOOLS

Our framework instantiates *seven* tools: four analysis tools for region discovery, two whole-image editors for global changes, and one region-level editor for localized edits.

Analysis Tools identify editable regions: (i) **SAM-2 + Qwen-3VL** (Ravi et al., 2024) for semantic segmentation with masks and descriptions; (ii) **DeepSeek-OCR** (Wei et al., 2025) for layout and text detection; (iii) **Qwen-Layered** (Yin et al., 2025) for foreground-to-background layer decomposition, capturing structural regions object-level segmentation may miss; (iv) **Qwen-BBox** (Bai et al., 2025) for instruction-guided bounding boxes, useful for adding or modifying objects not easily captured by image-only analysis.

Whole-Image Editing Tools. (v) **Qwen-Image-Edit** (Wu et al., 2025a) and (vi) **Flux-Kontext-Edit** (Labs et al., 2025) apply instruction-guided edits to the full image.

Region-Level Editing Tools. (vii) **Flux-Inpaint** (Labs et al., 2025) performs masked diffusion editing on regions from an analysis tool.

Whole-image tools operate directly; region-level tools need a prior analysis step and a valid region index. Allowed compositions are: (1) Layered/BBox/SAM-2/OCR $\rightarrow$ Flux-Inpaint; (2) Qwen-Image-Edit standalone; (3) Flux-Kontext-Edit standalone. All tools return structured JSON for consistent orchestration; see Appendix C.1.

4 EXPERIMENTS

Implementation Details. The planner and orchestrator are initialized from Qwen3-VL-8B (Bai et al., 2025) and fine-tuned with LoRA (Hu et al., 2022). The planner uses a lightweight setup ($r = 1$) on `q_proj` and `v_proj`, while the orchestrator uses higher capacity ($r = 64$) for flexible tool selection. Both train with learning rate 2×10^{-4}, $\alpha = 2r$, and batch size 16 on a single node of 8 A100 80GB GPUs.

Datasets. We first use images from MadVerse (Sagar et al., 2024), a large-scale multilingual advertisement dataset, and generate three abstract editing tasks per image with GPT-5, each requiring multi-step transformations such as cultural adaptation, audience retargeting, promotional shifts, product substitution, or stylistic changes. The orchestrator is trained on 6,685 image–instruction pairs and tested on 200 editing requests. The splits are *fully disjoint*—no shared source advertisements—so evaluation requires generalizing to unseen contexts with no image leakage. Since MadVerse adaptation tasks have no ground-truth edits, no baseline can be fine-tuned on this data; we therefore compare against strong open editors (FLUX.1-Kontext-dev, Qwen-Image-Edit-2511) under high-level, dense, and step-wise prompt settings (Tables 1 and 3). We also evaluate on GEdit-Bench (Liu et al., 2025a) and MagicBrush (Zhang et al., 2023), summarized in Sec. 4.3 with full tables in Appendix B.1.

4.1 MAIN RESULTS: COMPARISON TO END-TO-END EDITING BASELINES

We first compare to recent state-of-the-art open-source editors, testing whether they can reason about multi-step modifications and execute them in a single editing pass from a high-level instruction.

Baselines. We compare against FLUX.1-Kontext-dev (Labs et al., 2025) and Qwen-Image-Edit-2511 (Wu et al., 2025a) in two settings: (i) the high-level instruction is given directly, testing single-step reasoning and execution; and (ii) our base Qwen3-VL-8B first decomposes the task into simpler steps, all passed to the editor at once, testing whether a general MLLM's plan can be executed in a single-shot edit.

Method	Instruction Following	Identity Preservation	Visual Quality
FLUX.1-Kontext-dev (High-Level Instruction)	2.32	**4.32**	**3.525**
FLUX.1-Kontext-dev (Qwen3-VL-8B Plan)	3.33	3.005	2.405
Qwen-Image-Edit-2511 (High-Level Instruction)	3.355	2.769	2.26
Qwen-Image-Edit-2511 (Qwen3-VL-8B Plan)	3.807	3.005	2.16
Ours	**4.196**	3.155	2.525

Table 1: Comparison with end-to-end editing baselines, using averaged Gemini-3-Pro scores. Baselines either preserve the original but fail the instruction, or execute the edit at the cost of identity. Our method achieves the strongest instruction execution while maintaining good identity preservation.

Evaluation. A successful edit should follow the instruction, preserve important elements of the input, and look visually clean. We score each with Gemini-3-Pro (Google DeepMind, 2026) rated 1–5 from the input image, edited image, and high-level task; visual quality is instruction-agnostic and judged from the edited image alone. Using a different judge than in training, and corroborating with human A/B studies, keeps improvements from reflecting judge-specific artifacts. Scoring Instruction Following, Identity Preservation, and Visual Quality with a strong MLLM follows standard practice (e.g., GEdit-Bench (Liu et al., 2025a), EditScore (Luo et al., 2025)); we quantify its agreement with human ratings in Sec. 4.2 and give further details in Appendix C.3.1.

Results. Our method attains the highest instruction-following score, showing the benefit of explicit planning and step-by-step execution (Table 1). FLUX.1-Kontext-dev (High-Level Instruction) scores higher on identity preservation and visual quality, but mainly because it often leaves the image nearly unchanged, as reflected in its low instruction-following score (Fig. 6). Our method instead performs the requested edits while maintaining strong preservation and quality.

User Studies. To corroborate the judge results, we run a randomized A/B study: participants see paired results in random order and pick their preferred edit or a tie, weighing instruction following, identity preservation, and visual quality, with three raters per pair. Fig. 4 shows raters consistently prefer our edits over the baselines, underscoring the benefit of long-horizon planning and reward-driven orchestration. Study includes 80 participants (ages 18–84, mean 41.0; 47 male, 33 female; self-reported ethnicity: White 54, Mixed 9, Asian 7, Black 7, Other 3). Full protocol in Appendix C.3.1.

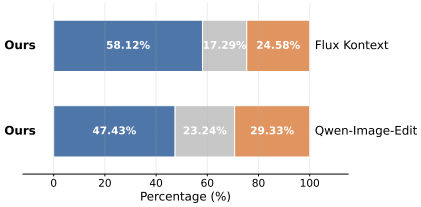

Figure 4: **User study (randomized A/B testing).** Results show a consistent human preference for our approach.

4.2 Reliability and Robustness of the VLM Judge

Optimizing a policy against a VLM judge risks reward hacking or judge-specific artifacts rather than real improvement. Several complementary checks indicate our reward signal is reliable. **(1) Alignment with human preference.** Human A/B preferences (Fig. 4) match the judge's rankings over FLUX.1-Kontext-dev and Qwen-Image-Edit-2511, and forced-choice Gemini judgments agree with humans in 67.56% of cases, essentially the 69.16% inter-annotator agreement—the judge matches humans about as often as they match each other. **(2) Independent reward model.** Re-deriving rewards with a model fine-tuned on human-verified edit pairs (UnifiedReward-Edit-qwen35-27b, trained on EditReward) gives consistent conclusions; we report the results in Appendix B.4. **(3) No judge fine-tuning.** The judge is chosen qualitatively and *not* fine-tuned on our data, so it cannot overfit task-specific idiosyncrasies. **(4) Decoupled judges.** Training rewards come from a GPT-based judge while all evaluations use a *different* model, Gemini-3-Pro, so gains cannot come from exploiting one scoring function. **(5) Score stability.** Over 100 image pairs on 5 runs, the average standard deviation is only 0.3445 (on the 1–5 scale).

4.3 Generalization to Standard Editing Benchmarks

Beyond abstract advertisement editing, we test generalization on standard *unseen* benchmarks disjoint from our training data. Our out-of-domain gains show the learned planning-and-orchestration policy transfers beyond advertisements rather than overfitting.

Method	Instr (%)	VQ
FLUX.1-Kontext-dev (sequential)	53.3	2.05
Qwen-BBox + FLUX Inpaint	60.0	2.105
Qwen-Image-Edit-2511 (sequential)	61.9	2.29
Qwen3VL-Base (k=5)	61.6	2.445
Ours (k=1)	63.9	2.245
Ours (k=3)	71.8	2.38
Ours (k=5)	**74.0**	**2.525**

Table 3: Instruction satisfaction (Instr), the percentage of checklist constraints satisfied, and Visual Quality (VQ). More inference branches improve both, and training the orchestrator gains 12.4% over the base model.

GEdit (Liu et al., 2025a) contains diverse edits including background changes, object-level modifications, and text editing. We compare against instruction-guided editors—InstructPix2Pix (Brooks et al., 2023), MagicBrush (Zhang et al., 2024a), AnyEdit (Yu et al., 2025), OmniGen (Xiao et al., 2025), OmniGen2 (Wu et al., 2025b), and Step1X-Edit (Liu et al., 2025a) with its improved Step1X-Edit-v1.1—and recent multimodal/generation systems capable of instruction-based edits: Qwen-Image-Edit (Wu et al., 2025a), UniWorld-v1 (Lin et al., 2025), Gemini 2.0 (Google DeepMind, 2025), BAGEL (Deng et al., 2025), FLUX.1 Kontext (Labs et al., 2025), and GPT-Image-1 (OpenAI, 2025).

We follow GEdit's VIEScore protocol (Ku et al., 2024), where an MLLM scores Semantic Consistency (SC) and Perceptual Quality (PQ), with overall score $\sqrt{\text{SC} \times \text{PQ}}$. As in GEdit, we use GPT-4.1 as the judge. Table 2 shows that our method achieves the best SC, PQ, and overall scores, demonstrating that the learned orchestrator generalizes beyond advertisement editing to broader benchmarks.

On **MagicBrush** (Zhang et al., 2023), ours improves CLIPScore by $+0.0189$ / $+0.0099$ over multi-tool agentic baselines GenArtist (Wang et al., 2024b) / LayerCraft (Zhang et al., 2025). Full tables, baselines, settings are in Appendix B.1.

Method	SC	PQ	Overall
AnyEdit	3.053	5.882	2.854
Instruct-Pix2Pix	3.296	6.189	3.219
MagicBrush	4.517	6.371	4.185
UniWorld-v1	4.93	7.43	4.85
OmniGen	5.879	5.871	5.005
Gemini 2.0 (DeepMind)	6.73	6.61	6.32
OmniGen2	7.16	6.77	6.41
Step1X-Edit	7.131	6.998	6.444
BAGEL	7.36	6.83	6.52
FLUX.1 Kontext [Pro]	7.02	7.60	6.56
Step1X-Edit-v1.1	7.658	7.354	6.969
GPT Image 1 [High]	7.85	7.62	7.53
Qwen-Image Edit	8.000	7.860	7.560
Ours	**8.153**	**8.030**	**7.604**

Table 2: Results on the GEdit benchmark. SC is semantic consistency, PQ perceptual quality, and Overall their geometric mean ($\sqrt{\text{SC} \times \text{PQ}}$). Our method stays on top even against recent proprietary and open-source models.

4.4 Orchestrator Design Choices

We isolate three design choices behind the Stage 2 orchestrator: (i) using multiple tools versus a single tool, (ii) training an explicit tool-calling policy versus prompting the base model training-free, and (iii) the effect of the number of inference candidates k.

Experiment Details. We fix our planner's multi-step plan across all variants and compare: (i) *FLUX.1-Kontext-dev (sequential)*: applied once per plan instruction; (ii) *Qwen-Image-Edit-2511 (sequential)*: same, invoked sequentially; (iii) *Qwen-BBox + FLUX Inpaint*: per step, Qwen-BBox proposes regions, a base Qwen3-VL selects one, and FLUX Inpaint applies the edit locally; (iv) *Base-Orchestrator*: an untrained Qwen3-VL selects tools with no learned policy, given the same tool-describing system prompt as our model. For (iii)'s effect of k, we evaluate $k = 1, 3, 5$.

Evaluation. Visual quality uses the same metric as before. Since all models receive the same plan, we evaluate instruction following and identity preservation more finely: GPT-5 generates constraints from the input image, task, and plan, specifying what to preserve, modify, or add and relevant attributes such as placement, orientation, and color. Gemini-3-Pro (Google DeepMind, 2026) then checks each constraint against the edited image. We report the percentage satisfied; see Appendix C.3.2.

As shown in Table 3, our trained orchestrator significantly outperforms all single-tool baselines and the untrained orchestrator, showing the benefit of both tool use and learning the selection policy from experience. Our multi-branch variants reach the highest instruction satisfaction and visual quality, and increasing k helps consistently: instruction satisfaction rises from 63.9% (k=1) to 71.8% (k=3) and 74.0% (k=5), with matching visual-quality gains, indicating that broader orchestration search yields more constraint-compliant, higher-quality edits.

4.5 PLAN DATASET: SELF- VS. EXTERNAL SUPERVISION

We test whether on-policy self-distillation improves plan quality over direct teacher imitation. An alternative to our checklist-guided self-training (Sec. 3.1) is to fine-tune the base model on plans from a stronger teacher LLM, but this risks distribution shift if teacher plans deviate from the base model's native patterns. To quantify this, we measure the base model's perplexity under teacher forcing on (1) checklist-conditioned plans from the base model itself and (2) plans from a teacher LLM (GPT-5 (OpenAI, 2026)). Teacher plans yield far higher perplexity (61.25 vs. 4.89, lower is better), placing them well outside the base model's intrinsic distribution and evidencing potential instability under off-policy imitation. Our checklist-guided self-distillation instead keeps supervision on-policy and aligned with the model's natural generation.

4.6 EMPIRICAL VALIDITY OF THE REWARD APPROXIMATIONS

Our training relies on two approximations (Sec. 3.2): additive rewards and original-image independence. Since these decouple sequential edits, we probe the original-image independence approximation directly.

Using verifier-guided selection, the best tool per subtask under *ideal sequential execution* achieves an average reward of 4.1873. Evaluating these same tools on the *original* image, as done during precomputation, yields 3.4057, compared with 1.8688 for an arbitrary tool. Thus, original-image evaluation remains a strong proxy for sequential execution and preserves tool rankings sufficiently well for orchestrator training.

Large reward gaps between applying the same tool to the *true intermediate image* versus the *original image* (> 2.5) occur for only 8.3% of subtasks, consistent with checklist-guided plans typically targeting distinct regions. These cases are further mitigated by closed-loop refinement (Sec. 3.3), which prunes systematically infeasible subtasks, and the inference-time verifier (Sec. 3.4), which re-ranks tools on the *true* intermediate image. We therefore treat this approximation as a tractability trade-off rather than an independence assumption.

4.7 QUALITATIVE RESULTS

Figures 6, 7, and 8 show our method on diverse long-horizon advertisement edits requiring coordinated updates to visuals, text, layout, and branding. We blur faces to preserve anonymity.

Long-Horizon Adaptation. Figs. 6, 7 present challenging transformations (e.g., adapting for business travelers or US Independence Day) with coupled changes to background, palette, slogans, badges, and branding. Single-step editors often produce partial or excessive modifications that break layout or identity, whereas ours stays globally coherent, instruction-faithful, and brand-consistent.

Planner–Orchestrator Decomposition. Fig. 8 shows how checklist-guided planning breaks abstract instructions into atomic subtasks executed sequentially via reward-driven orchestration, giving comprehensive coverage and coherent multi-step transformations rather than isolated local edits.

Additional step-by-step visualizations are in Appendix A.

5 CONCLUSION

We presented an experiential framework for long-horizon, open-ended image editing that pairs checklist-guided on-policy planning with reward-driven orchestration, moving beyond handcrafted pipelines and single-step generation. The planner learns structured decompositions, the orchestrator selects tools and regions from outcome-based feedback, and closed-loop refinement aligns plans with executable actions. Experiments and user studies show our method produces more coherent, instruction-faithful edits than strong single-step and rule-based multi-step baselines. We further analyze the cost of our approximations, characterize the VLM judge's reliability, and show that gains transfer to unseen benchmarks and hold under an open-source reward model.

Limitations. Our method costs more at inference than single-step editors (Table 10), and the additive / original-image-independence approximations, though accurate for most subtasks, break down for strongly coupled global edits (Sec. 4.6). Cutting inference latency and modeling residual subtask coupling are promising future directions.

AI USE STATEMENT

In this work, we used generative AI tools for coding, polishing our writing, finding relevant work, as well as synthetic data creation. We have not used generative AI tools for research ideation, proving mathematical claims or drafting sections of the paper. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness, and the provided related work was carefully checked. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work improves the ability of automated systems to perform complex, multi-step image edits. While this can support benign creative and design workflows, the same capability may be misused to produce misleading, manipulative, or unauthorized visual content. In particular, tasks involving audience or cultural adaptation may reflect or amplify biases and stereotypes present in the underlying vision-language and image editing models. We therefore view appropriate human oversight, provenance mechanisms, and safeguards against harmful or deceptive use as important for deployment. Our experiments use existing datasets and pretrained models under their respective terms, and trademarks and copyrighted images are used only for research and identification purposes, with no affiliation or endorsement implied. Our human evaluation consists of randomized A/B preference judgments, and no personally identifying information is used in our analysis.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our method and experiments throughout the main paper and supplementary material. The model architecture, training objectives, and planner and orchestrator formulations are described in Sec. 3. Dataset construction, task definitions, evaluation protocols, and implementation details are provided in Sec. 4 and the Appendix, including prompts, tool configurations, hyperparameters, and additional ablations. We also include qualitative examples and detailed descriptions of the editing tools and region proposal mechanisms used in our experiments.

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.

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

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22560–22570, 2023.

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.

Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. *arXiv preprint arXiv:2309.17102*, 2023.

Google DeepMind. Gemini 2.0. `https://gemini.google.com/`, 2025. Accessed: 2026-03-12.

Google DeepMind. Gemini 3 Pro, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.

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.

Amir Hertz, Andrey Voynov, Shlomi Fruchter, and Daniel Cohen-Or. Style aligned image generation via shared attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4775–4785, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *arXiv preprint arXiv:2406.09403*, 2024.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

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

Liya Ji, Chenyang Qi, and Qifeng Chen. Instruction-based image editing with planning, reasoning, and generation. In *ICCV*, 2025.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. Viescore: Towards explainable metrics for conditional image synthesis evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12268–12290, 2024.

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, 2025. URL https://arxiv.org/abs/2506.15742.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22511–22521, 2023.

Bin Lin, Zongjian Li, Xinhua Cheng, Yuwei Niu, Yang Ye, Xianyi He, Shenghai Yuan, Wangbo Yu, Shaodong Wang, Yunyang Ge, et al. Uniworld-v1: High-resolution semantic encoders for unified visual understanding and generation. *arXiv preprint arXiv:2506.03147*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024.

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.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025b.

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.

Shichao Ma, Yunhe Guo, Jiahao Su, Qihe Huang, Zhengyang Zhou, and Yang Wang. Talk2image: A multi-agent system for multi-turn image generation and editing. *arXiv preprint arXiv:2508.06916*, 2025.

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.

Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. *arXiv preprint arXiv:2307.02421*, 2023.

Weili Nie, Sifei Liu, Morteza Mardani, Chao Liu, Benjamin Eckart, and Arash Vahdat. Compositional text-to-image generation with dense blob representations. *arXiv preprint arXiv:2405.08246*, 2024.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2025-05-13.

OpenAI. Introducing 4o image generation. https://openai.com/index/introducing-4o-image-generation/, 2025. Accessed: 2026-03-12.

OpenAI. ChatGPT, 2026.

Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–11, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Amruth Sagar, Rishabh Srivastava, Venkata Kesav Venna, Ravi Kiran Sarvadevabhatla, et al. Madverse: A hierarchical dataset of multi-lingual ads from diverse sources and categories. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 8087–8096, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8839–8849, 2024.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11888–11898, 2023.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2507.18624.

Xudong Wang, Trevor Darrell, Sai Saketh Rambhatla, Rohit Girdhar, and Ishan Misra. Instancediffusion: Instance-level control for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6232–6242, 2024a.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. Genartist: Multimodal llm as an agent for unified image generation and editing. *Advances in Neural Information Processing Systems*, 37: 128374–128395, 2024b.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025a. URL https://arxiv.org/abs/2508.02324.

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*, 2025b.

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*, 2025c.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13294–13304, 2025.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

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 *ICML*, 2024.

Chun-Hsiao Yeh, Yilin Wang, Nanxuan Zhao, Richard Zhang, Yuheng Li, Yi Ma, and Krishna Kumar Singh. Beyond simple edits: X-planner for complex instruction-based image editing. *arXiv preprint arXiv:2507.05259*, 2025.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, Lionel M. Ni, Jingren Zhou, Junyang Lin, and Chenfei Wu. Qwen-image-layered: Towards inherent editability via layer decomposition, 2025. URL https://arxiv.org/abs/2512.15603.

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.

Ziyun Zeng, Hang Hua, and Jiebo Luo. Mira: Multimodal iterative reasoning agent for image editing. *arXiv preprint arXiv:2511.21087*, 2025.

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.

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, 2024a.

Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, et al. Hive: Harnessing human feedback for instructional visual editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9026–9036, 2024b.

Yuyao Zhang, Jinghao Li, and Yu-Wing Tai. Layercraft: Enhancing text-to-image generation with cot reasoning and layered object integration. *arXiv preprint arXiv:2504.00010*, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Wang Zhenyu, Li Aoxue, Li Zhenguo, and Liu Xihui. Genartist: Multimodal llm as an agent for unified image generation and editing. *NeuRIPS*, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

APPENDIX

We provide additional qualitative results, experimental comparisons, and implementation details below.

A QUALITATIVE RESULTS

We provide step-wise editing visualizations in Tables 4, 5, and 6. These examples illustrate our method's ability to execute long sequences of diverse edits, including text changes, background modifications, and object-level alterations. Together, they demonstrate that the system can reliably compose multiple heterogeneous edits while maintaining visual coherence and consistency with the original content.

B ADDITIONAL EXPERIMENTS

B.1 RESULTS ON IMAGE EDITING BENCHMARKS

In the main paper, we primarily reported results on a dataset consisting of complex multi-step advertisement editing tasks. In this section, we further evaluate our method on several widely used image editing benchmarks. Specifically, we consider the multi-turn editing setting of MagicBrush (Zhang et al., 2024a) as well as the single-turn editing setting introduced in GEdit-Bench (Liu et al., 2025a).

B.1.1 PERFORMANCE ON MAGICBRUSH

The multi-turn editing setting in MagicBrush consists of a sequence of edits (typically ranging from one to three) applied to a single image. The instructions in this benchmark specify direct edits, rather than requiring high-level planning. Therefore, in this experiment we employ only our learned orchestrator and do not use the planner. Through this evaluation, we examine whether the policy learned by our orchestrator—trained on advertisement editing tasks—generalizes to other commonly studied image editing scenarios.

Baselines We compare our method against several existing agentic image editing systems, including GenArtist (Wang et al., 2024b), LayerCraft (Zhang et al., 2025), and Talk2Image (Ma et al., 2025). In addition to these agent-based pipelines, we also evaluate against widely used instruction-based editing models such as MagicBrush (Zhang et al., 2024a), HIVE (Zhang et al., 2024b), and InstructPix2Pix (Brooks et al., 2023).

Unlike our approach, these agentic baselines do not learn a tool-selection policy from experience; instead, they rely on prompt-engineered orchestration to select the appropriate actions.

Evaluation The MagicBrush benchmark conventionally provides five evaluation metrics. CLIP-T measures text alignment by comparing the CLIP embedding of the generated image with the CLIP embedding of the target caption. The remaining metrics—L1 distance, L2 distance, CLIP-I, and DINO—measure similarity between the generated image and a ground-truth image created by human annotators, conditioned on a target mask. However, due to the open-ended nature of image editing tasks, the same instruction can often be satisfied in multiple visually valid ways.

We attach illustrative examples in Fig. 5. In both examples, the edits produced by our method correctly follow the sequence of instructions while preserving the overall scene structure. However, the resulting images differ from the single ground-truth image provided in the benchmark.

For instance, in the first example, our method successfully applies strawberry glaze to the donuts, replaces the alcohol shelf with a bookshelf, and places a smiling person nearby. While the specific appearance of these elements differs from the ground truth (e.g., the exact pose of the person or the texture of the glaze), the requested transformations are clearly satisfied. Similarly, in the second example, our method correctly removes the sunglasses, replaces the glove with a wooden block, and adds a small action figure beside it. Although the precise visual realization differs from the reference image, the instruction is executed faithfully.

Execution Result	Plan
	**Goal:** Adapt the advertisement into a vibrant millennial-focused savings campaign while maintaining brand recognition.
	**1:** Replace the background with a vibrant gradient of orange, teal, and purple, keeping the jar of coins and plant unchanged. flux kontext edit
	**2:** Replace the text 'Public Provident Fund (PPF) Scheme' with 'Start Saving Today' in bold, modern sans-serif font, centered in the top-left area. qwen image edit
	**3:** Replace the text 'The Scheme offers an investment avenue...' with 'Grow Your Future, Even While You're Just Starting Out!' in bold, modern sans-serif font. qwen-layered (region3) + flux inpaint
	**8:** Add a subtle glow effect around the jar of coins and plant to draw attention to them while preserving all other elements. qwen image edit
	**9:** Include a small, clean smartphone screen graphic in the bottom-right corner showing a simple savings app interface to reinforce digital accessibility. qwen-layered (region1) + flux inpaint

Table 4: Example editing sequence showcasing the transition from the original advertisement to the final millennial-focused design. Our planner decomposes the task into steps, and the orchestrator carries out each step by selecting the appropriate editing tools.

Execution Result	Plan
	**Goal:** Adapt the advertisement into a localized version targeted at spice lovers in Mexico.
	**1:** Add a spicy dish on the right side of the image, placing it next to the product, without altering the product or existing text. qwen-bbox (region3) + flux inpaint
	**3:** Add a small image of a traditional Mexican chili pepper, such as a jalapeño or habanero, near the bottom center of the image, below the product qwen-bbox (region2) + flux inpaint
	**4:** Replace 'chilli eating competition' with 'spice challenge' in the black banner while preserving design elements. deepseek-ocr (region4) + flux inpaint
	**6:** Change the background to a vibrant red with subtle spice-themed patterns (chili outlines/smoke), keeping product and text in place. qwen-layered (region1) + flux inpaint
	**7:** Add bold white text 'Perfect for Mexican Spicy Dishes' at the top of the image, positioned above the 'TAG A FRIEND' headline. qwen image edit

Table 5: Example editing sequence showcasing the transition from the original advertisement to a localized version targeted at spice lovers in Mexico. Our planner decomposes the task into steps, and the orchestrator carries out each step by selecting the appropriate editing tools.

Execution Result	Plan
	**Goal:** Create a healthy lifestyle version targeting fitness-conscious buyers.
	**2:** Change the color palette to light blue, green, and earthy tones across the entire image, including the noodles bowl, packaging, and background. qwen-bbox (region1) + flux inpaint
	**4:** Add a gym locker room background or gym setting behind the noodles bowl, without obscuring the food or packaging. qwen-bbox (region2) + flux inpaint
	**5:** Add the tagline 'Fuel Your Fitness Journey' in bold, modern font directly below the main slogan 'Jhatpat BNAO BEFIKAR Khao', keeping the existing font style and placement intact. qwen image edit
	**6:** Add the phrase 'Perfect for Pre-Workout Fuel' next to the noodle bowl on the left, using the same font as the existing slogan. qwen image edit
	**7:** Replace the existing decorative elements with a sports-themed background (e.g., crossed dumbbells or a 'Grab a Pair' prompt in front of dumbbells) to reinforce the fitness theme.' qwen-layered (region1) + flux inpaint

Table 6: Example editing sequence showcasing the transition from the original advertisement to a healthy lifestyle focused version. Our planner decomposes the task into steps, and the orchestrator carries out each step by selecting the appropriate editing tools.



Figure 5: Examples from the MagicBrush benchmark. Each example shows the input image, the sequence of editing instructions, the provided ground-truth result, and the output produced by our method. While the benchmark provides a single ground-truth image for evaluation, many editing instructions can be satisfied in multiple visually valid ways. As illustrated here, our method produces edits that follow the instructions while maintaining visual coherence, but may differ from the specific appearance of the provided ground truth. This highlights a limitation of evaluation metrics that rely on similarity to a single reference image when assessing open-ended image editing tasks.

Baseline	**CLIP-T**
Instruction-based editing models	
InstructPix2Pix	0.2726
w/ MagicBrush	0.2754
HIVE	0.2673
w/ MagicBrush	0.2796
Agentic pipelines	
GenArtist	0.3067
LayerCraft	0.3157
Talk2Image	0.3157
Ours (k=5)	**0.3256**

Table 7: Performance on the MagicBrush benchmark measured using CLIP-T. Our method outperforms prior agentic approaches (GenArtist, LayerCraft, and Talk2Image) as well as instruction-based editing approaches (InstructPix2Pix, HIVE, MagicBrush).

These examples highlight a limitation of metrics that rely on similarity to a single ground-truth image: multiple visually valid edits may satisfy the instruction, yet still receive a lower score due to differences in appearance. As a result, such metrics may penalize correct edits that deviate from the specific reference provided in the dataset. For this reason, we focus on measuring alignment with the textual description, as captured by the CLIP-T score.

Results We report the CLIP-T results in Table 7. Our method achieves the best performance, outperforming both prior agentic approaches and instruction-based editing models in terms of semantic alignment with the target instructions. The policy learned through experience enables our orchestrator to select the tools best suited for each task, resulting in minimal yet accurate edits to the original image. This highlights the advantage of a learning-based orchestration strategy, which allows the

Comparison	Gemini Preference
Checklist Planner vs Base Qwen3-VL	**70.25%**

Table 8: Pairwise plan preference measured using Gemini-3-Pro. Each pair of plans is evaluated twice with reversed ordering to mitigate positional bias.

Method	Instruction Following	Identity Preservation	Visual Quality
GenArtist	1.252	1.007	1.660
Ours	**4.196**	**3.155**	**2.525**

Table 9: **Comparison with the agentic editing baseline GenArtist.** Scores are averaged Gemini-3-Pro evaluations measuring instruction following, identity preservation, and visual quality. GenArtist often introduces excessive degradation to the original image while attempting edits, resulting in low scores across all metrics. In contrast, our method performs the requested edits while maintaining substantially stronger identity preservation and overall visual quality.

system to adapt tool usage based on experience rather than relying on heuristic, prompt-engineered orchestration strategies.

B.2 Are Checklist-Based Plans Better?

In this section, we investigate whether checklist-based supervision leads to higher-quality plans compared to directly generating plans with a base model. While the final edited images produced by executing these plans provide one signal of performance, they do not directly indicate which plan reflects a deeper understanding of the task itself.

To study this, we compare two types of plans: (1) plans generated by our planner trained with checklist-based supervision, and (2) plans generated by a base Qwen3-VL model that is simply prompted to produce a plan for adapting the image, without being required to satisfy any explicit constraints.

Evaluation. Evaluating the quality of plans is inherently subjective. Therefore, we rely on a strong MLLM judge, Gemini-3-Pro (Google DeepMind, 2026), to compare pairs of plans and select the better one. To mitigate positional bias (e.g., a tendency to favor either the first or second option in pairwise comparisons), we evaluate each pair of plans twice: once with the original ordering and once with the order reversed. The final preference score is computed by averaging the outcomes across both evaluations.

Results. Table 8 reports the pairwise preference results. Gemini prefers the checklist-based plan over the base model plan in **70.25%** of comparisons. This result suggests that checklist supervision encourages the planner to produce more coherent and task-aware editing strategies.

B.3 GenArtist Performance on Advertisement Editing

In the main paper, we compare our method with several agentic baselines, including Qwen Image Edit and FLUX Kontext, with an external MLLM for planning. In this section, we additionally evaluate a widely used open-source agentic editing system, GenArtist (Wang et al., 2024b) on our MadVerse image-based benchmark. Other recent agentic approaches such as X-Planner (Yeh et al., 2025) and MIRA (Zeng et al., 2025) do not provide publicly available implementations, making direct comparisons difficult. We therefore focus on GenArtist as the strongest reproducible baseline and evaluate it under the same experimental setting described in Sec. 4.1. We report our results in Table 9. We observe that GenArtist struggles to effectively execute the requested edits compared to our method, often resulting in significant degradation. We hypothesize that this may stem from certain tools being poorly suited for these tasks, as well as limitations in the orchestrator's policy.

Method	Time	Granularity
FLUX.1-Kontext-dev	29.4s	per prompt
Qwen-Image-Edit-2511	40.6s	per prompt
Ours (k=1)	66.01s	per subtask (5–10/edit)
Ours (k=3)	77.99s	per subtask (5–10/edit)

Table 10: Inference-time cost. Single-step baselines edit in one pass, whereas our method runs 5–10 subtasks per edit with k branches each and verifier re-ranking. We trade speed for the ability to complete abstract, long-horizon edits that the baselines cannot (Tables 1, 3).

Setting	Avg. Max Reward
Before filtering	4.1708
After filtering	4.3095

Table 11: Average maximum achievable sub-task reward (Gemini 3 Pro judge) before and after plan refinement.

B.4 Robustness to the Choice of Reward Model

Our framework is agnostic to the specific reward model, provided it is sufficiently strong. To verify that our results are not tied to a proprietary judge, we retrain the orchestrator using the open-source reward model UnifiedReward-Edit-qwen35-27b (trained on EditReward) in place of the GPT-based rewards, keeping everything else fixed. Relative to the GPT-reward orchestrator, Instruction Following changes by -0.295, while Identity Preservation and Visual Quality improve by $+0.295$ and $+0.16$, respectively—i.e., comparable overall performance with a slightly different trade-off. This confirms that the experiential learning procedure, rather than any single closed-source judge, drives the observed gains, and that a fully open-source instantiation of our pipeline is feasible.

B.5 Computational Analysis

Training cost. Naively scoring full editing trajectories is intractable: for a plan with L subtasks and B tool–region candidates per subtask, exhaustive trajectory evaluation requires $O(B^L)$ diffusion generations. Our additive and original-image-independence approximations let us evaluate each candidate independently per subtask, reducing this to $O(LB)$—linear in the number of subtasks—so all tool–region rewards can be precomputed once and reused during training.

Inference latency. Our pipeline explicitly trades speed for quality. Executing a single subtask takes 66.01s at k=1 and 77.99s at k=3, and a typical edit contains 5–10 subtasks, versus 29.4s (FLUX.1-Kontext-dev) and 40.6s (Qwen-Image-Edit-2511) for a single-pass edit (Table 10). This overhead is inherent to multi-step, multi-branch orchestration with verifier re-ranking. Crucially, the fast single-step baselines fail on these abstract tasks *even when given the same decomposed plan and executed sequentially* (Tables 1 and 3), and standard multi-tool agentic pipelines (e.g., GenArtist, X-Planner) are the appropriate comparison here; we outperform representative such systems (Sec. 4.3). The additional cost therefore buys edits the faster baselines cannot produce. We prioritize edit quality and view inference-speed optimization—e.g., subtask parallelization, verifier/branching reduction, and distillation—as future work.

B.6 Ablation Study: Effect of Plan Refinement

To quantify the impact of our closed-loop plan refinement (Sec. 3.3), we compute, for each subtask, the maximum achievable reward across all tools and regions, $\max_{a,r} R_{a,r}(s_t)$, and average this value over the samples. Rewards are assigned by a VLM-based judge.

Table 11 reports the average maximum reward before and after filtering infeasible sub-tasks. Filtering infeasible sub-tasks increases the average reward, quantifying improved compatibility between plans and executable tools.



Figure 6: **Qualitative results on diverse long-horizon advertisement editing tasks.** These examples show two challenging instructions—adapting for Business travelers, and for American Independence Day. Single-step editors (Flux Kontext (Labs et al., 2025) and Qwen-Image-Edit (Wu et al., 2025a)) often perform partial stylistic changes or introduce minimal or shallow modifications in text, layout, or branding. In contrast, our method consistently produces edits that are more instruction-faithful and globally coherent, jointly updating visual themes, textual content, layout elements, and brand messaging. These results demonstrate the effectiveness of checklist-guided planning and reward-driven orchestration in handling abstract, multi-step transformations beyond localized edits.

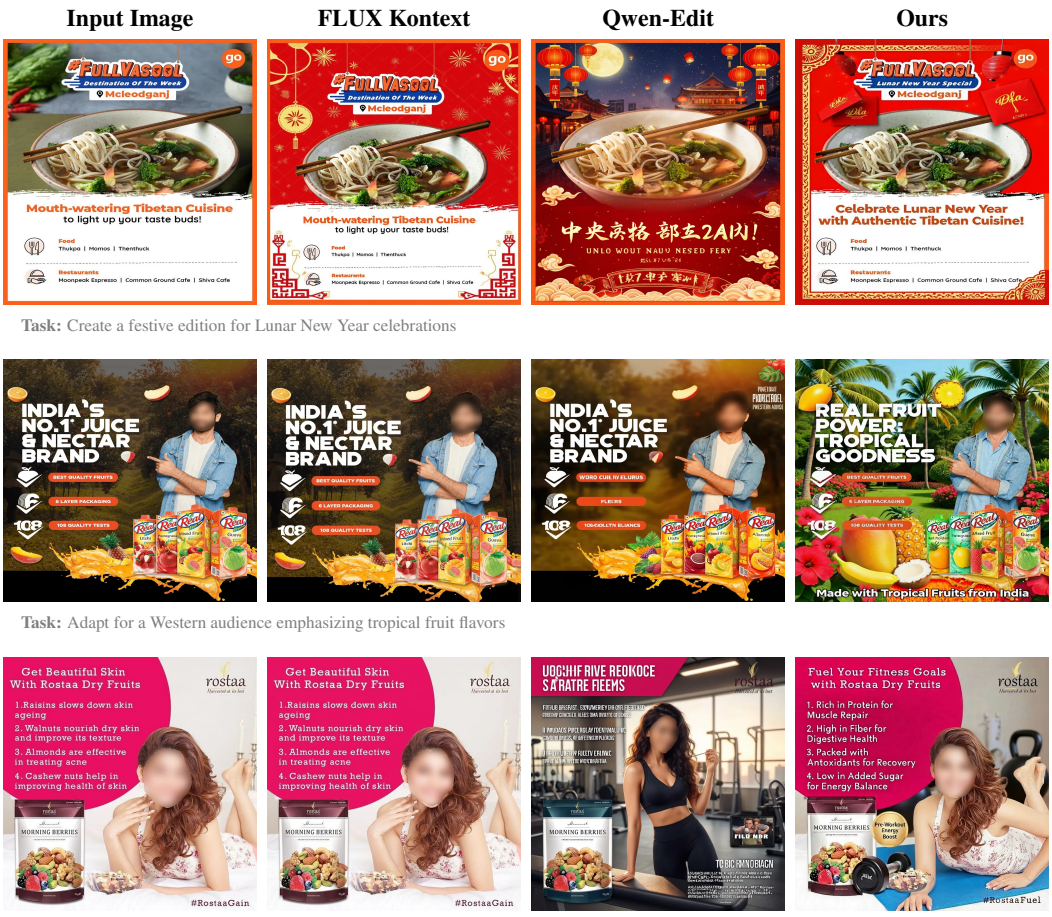

Figure 7: **Qualitative results on diverse long-horizon advertisement editing tasks.** Three challenging instructions—adapting for Lunar New Year, a western audience, and a fitness-conscious audience. Our method stays faithful to the instruction and globally coherent, jointly updating visual themes, text, layout, and brand messaging. Faces are blurred to preserve anonymity.

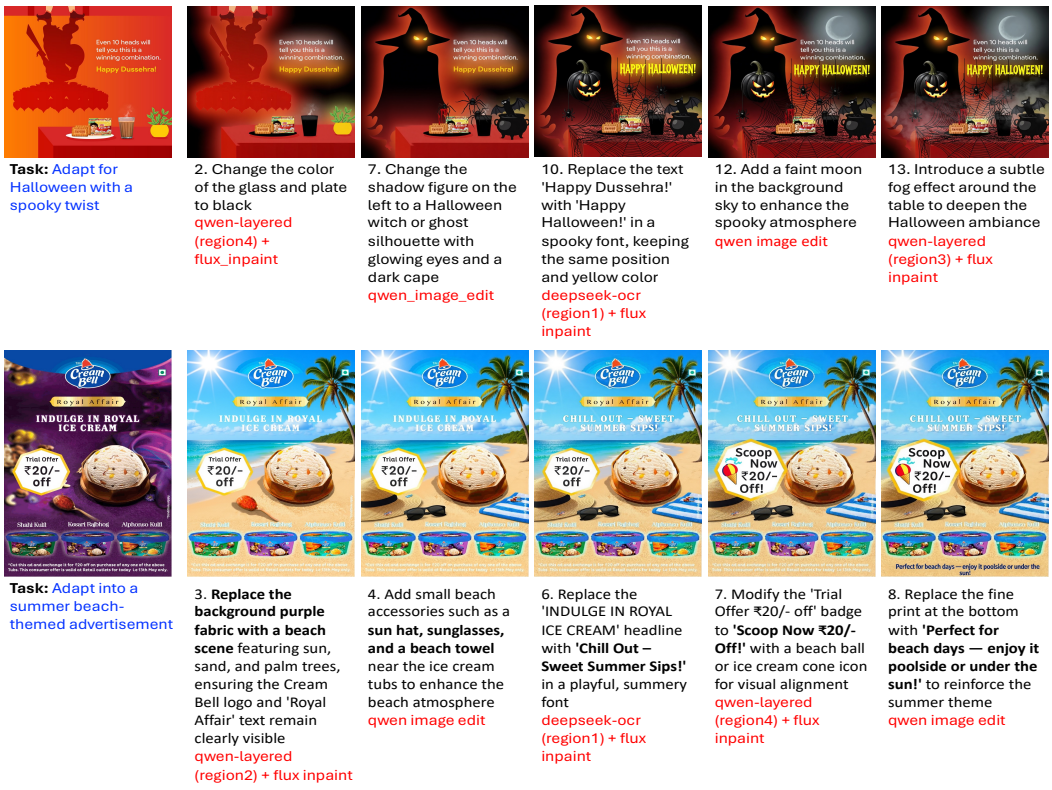

Figure 8: **Planner–orchestrator subtasks and their outputs.** Each row starts with the input advertisement and a high-level instruction (first column), followed by the sequence of subtask plans and edits. These show how checklist-guided planning decomposes an abstract instruction into atomic edits and how the orchestrator selects tool–regions to execute them into a coherent, instruction-faithful design. We show samples from the full sequence due to space.

C Implementation Details

Our framework consists of three main components: (i) a **Planner** that decomposes high-level editing requests into a sequence of atomic operations, (ii) a set of **editing tools** that perform the underlying image transformations, and (iii) an **Orchestrator** that selects the appropriate tool and/or spatial region to execute each operation. In this section, we primarily focus on the editing tools and the orchestrator used during execution.

For each component, we describe how it is constructed, trained, and used during inference. Finally, since the editing tasks we consider are open-ended, we also describe the evaluation framework used to assess instruction following, identity preservation, and visual quality. Finally, we also provide more details on our inference algorithm.

C.1 Editing Tools

Image editing tasks involve a wide range of transformations, from global changes such as modifying the background or color palette to localized edits such as replacing objects or modifying specific regions. No single model reliably supports all of these operations, motivating the use of multiple specialized editing tools.

We therefore employ a modular toolset consisting of three categories: *analysis tools*, *whole-image editing tools*, and *region-level editing tools*. Analysis tools identify regions of interest (e.g., objects, layout elements, or semantic layers). Whole-image editing tools apply instruction-guided transformations to the entire image. Region-level editing tools instead operate on specific regions identified by the analysis tools, enabling more precise localized edits.

C.1.1 WHOLE-IMAGE EDITING MODELS

For global edits, we use two instruction-guided editing models: *FLUX.1-Kontext-dev* and *Qwen-Image-Edit-2511*. Both take an input image and a textual instruction and generate a modified image that reflects the requested change while preserving the overall structure of the original image.

FLUX.1-Kontext-dev and **Qwen-Image-Edit-2511** are image editing models that take an input image and a textual instruction and generate an edited image consistent with the requested modification.

Both models provide strong instruction-following capabilities and produce high-quality edits, but they also exhibit certain weaknesses. In particular, since edits are conditioned on the full image, modifications are not strictly spatially constrained and may unintentionally affect regions unrelated to the intended change. To address this limitation, we additionally incorporate a region-level editing pipeline that first identifies relevant regions using analysis tools and then performs targeted modifications via diffusion-based inpainting.

C.1.2 ANALYSIS TOOLS

These tools identify editable regions in the input image that can later be modified by region-level editing models. Because different editing tasks require different forms of spatial understanding, we employ multiple complementary region discovery mechanisms to detect relevant areas of the image.

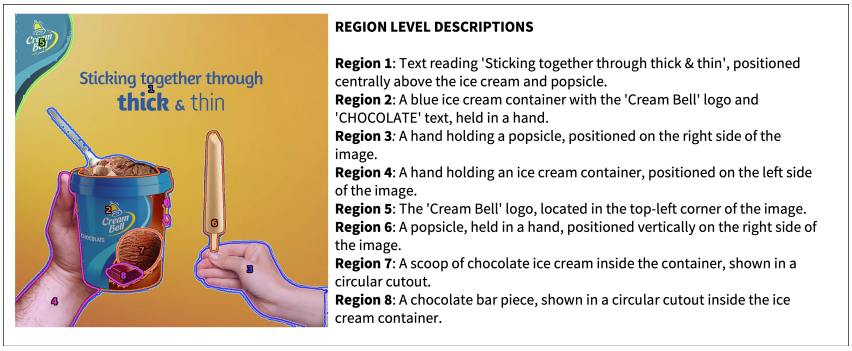

Figure 9: Region discovery using the SAM-2 + Qwen3-VL pipeline. SAM-2 first segments the image into candidate regions, which are visualized with numbered masks (left). The marked image is then provided to Qwen3-VL, which generates semantic descriptions for each region (right). These region indices and descriptions allow the system to reference specific parts of the image during subsequent editing steps.

(i) **SAM-2 + Qwen3-VL** performs semantic region discovery using a Set-of-Marks representation. We first apply SAM-2 to segment the image into candidate regions and overlay numbered markers on the resulting masks (Yang et al., 2023). The marked image is then provided to Qwen3-VL-8B, which generates a semantic description for each numbered region. This produces a structured mapping between region indices, masks, and textual descriptions, allowing the system to reference specific regions during editing. Fig. 9 shows an example. We consider up to eight candidate regions, selected based on the largest area.

(ii) **DeepSeek-OCR** performs layout and text detection, identifying bounding boxes corresponding to textual elements and structured layout regions. Fig. 10 shows an example. We consider up to 10 candidate regions, selected based on the largest area.

(iii) **Qwen-Layered** decomposes the image into a set of alpha-composable layers ordered from foreground to background, capturing larger structural components that may not correspond to individual objects. Each predicted alpha layer is converted into a binary mask by thresholding the alpha values at 128. These masks can then be used as candidate editable regions. Example layers are shown in Fig. 11. We consider four candidate regions.

(iv) **Qwen-BBox** addresses a limitation of the previous analysis tools. While SAM-2, DeepSeek-OCR, and Qwen-Layered are effective at identifying existing objects, text, or structural components,



Figure 10: DeepSeek-OCR identifies textual and layout elements in the image by predicting bounding boxes around text regions and structured layout components. Each detected region is assigned an index, allowing the system to reference and modify specific textual elements during editing.



Figure 11: Layer-based region discovery using Qwen-Layered. **Left:** Original input image. **Middle and Right:** Example masks produced by Qwen-Layered that highlight different structural regions of the scene. These masks correspond to different alpha-composable layers that can be independently edited in subsequent region-level editing steps.

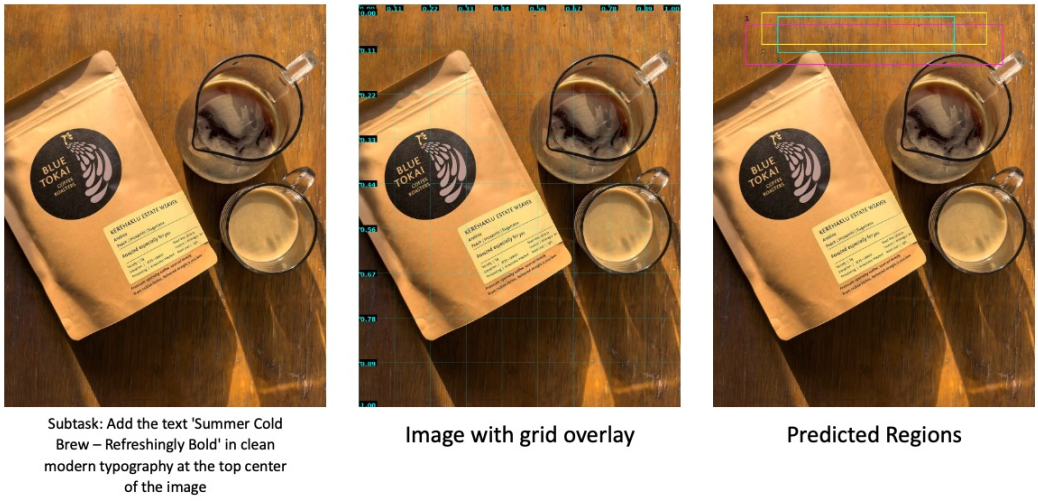

Figure 12: Instruction-guided region discovery using Qwen-BBox. **Left:** Original input image with the editing instruction. **Middle:** A grid overlay is added to the image, providing spatial reference cues that help the model reason about locations when predicting regions. **Right:** Candidate bounding boxes predicted by Qwen3-VL conditioned on the editing instruction. The model predicts three candidate regions that can be selected individually or combined during region-level editing.

they are not task-specific. As a result, they can fail to identify regions required for edits that involve adding new objects or modifying areas that do not correspond to clearly defined semantic entities.

To address this, we use a Qwen3-VL-8B model to directly predict candidate regions conditioned on the editing instruction. However, we observed that predicting absolute bounding box coordinates directly is difficult for the model. Instead, we parameterize bounding boxes using normalized coordinates expressed as percentages of the image width and height. Even with normalized coordinates, the model struggles to localize regions reliably without visual references. Therefore, we overlay a grid on the image (Fig. 12, middle), which serves as a visual prompt that helps the model reason about spatial locations.

Given the instruction and the grid-annotated image, the model predicts three candidate bounding boxes (Fig. 12, right). During editing, the system may choose to operate on any of these individual regions or on the union of all predicted boxes.

C.1.3 Region-Level Editing Tools

For region-level editing, the orchestrator first selects a target region from the outputs of the analysis tools. Each analysis tool proposes candidate regions (e.g., segmentation masks or bounding boxes), from which the system chooses a single mask corresponding to the intended edit. In the case of bounding boxes we convert the box to a binary mask.

Given the selected mask, we use **FLUX-Kontext Inpaint**, a diffusion-based editor that performs instruction-guided modifications within the specified region. The model takes as input the image, the textual instruction, and the binary mask, and generates edits that are constrained to the selected area while preserving the surrounding content.

To provide the editing model with greater flexibility when modifying the target object, we dilate the predicted mask by 100 pixels before applying inpainting. Expanding the mask allows the model to adjust the size, shape, or surrounding context of the edited region, rather than being strictly constrained to the original mask boundary. The masked region is then edited according to the instruction, while pixels outside the mask remain unchanged.

This region-level editing mechanism enables precise localized modifications that are difficult to achieve with whole-image editing models alone.

C.2 Orchestrator Details

Given the input image and editing instruction, the orchestrator selects the next action by producing a structured tool call. Each action is represented as a JSON object of the form

$$\{\texttt{"tool"} : t,\ \texttt{"arguments"} : a\},$$

where t denotes the selected tool and a contains any required parameters.

At each step, the orchestrator can choose between two types of actions: invoking an *analysis tool* or directly applying a *whole-image editing tool*. If a whole-image editing tool is selected, the model performs the requested modification across the entire image.

If an analysis tool is selected, the tool returns a set of candidate regions (e.g., segmentation masks or bounding boxes). These regions are then made available to the orchestrator, which subsequently selects one of them when invoking a region-level editing tool. In this case, the tool call includes both the editing instruction and the index of the region to be modified.

For example, a region-level edit is represented as

```
{
  "tool": "flux_inpaint",
  "arguments": {"region_number": 3}
}
```

C.2.1 Reward Model

Since the editing tasks we consider are open-ended and do not have a single fixed ground-truth target, it is difficult to directly determine whether a generated edit successfully satisfies the instruction. Therefore, we require a signal that evaluates the quality of candidate edits and allows the system to identify which tool execution performs best.

As discussed in the main paper, we pre-compute the outputs of candidate tool calls and use a reward model to score each resulting edit. The orchestrator is then trained to select the tool which yielded the highest reward.

Existing reward models such as EditScore (Luo et al., 2025) and EditReward (Wu et al., 2025c) are primarily designed for natural image editing and do not fully capture the requirements of other kinds of images e.g., advertisement-style edits. Therefore, we design a custom evaluation rubric based on three criteria: *instruction execution*, *identity preservation*, and *visual quality*. The rubric used by the evaluator is shown below.

CRITERION 1: INSTRUCTION EXECUTION

Question:
Did the edited image correctly and completely execute the requested instruction?

Scoring (0–5):

0: No attempt / wrong task: The instruction was not executed at all, or a completely unrelated edit was performed. The new advertisement does not actually advertise what the user intended to advertise.

1: Attempted but ineffective: The model made a relevant attempt (e.g., tried to change the text or background as instructed), but it did not succeed in completely executing any part of the instruction.

2: Partial but meaningful: At least one major part of the instruction is executed correctly, but other parts are missing or wrong.

3: Mostly right, clearly flawed: Mostly correct, but some part of the instruction has been completely misinterpreted. Such as text is placed below instead of above.

4: Complete with minor issues: All requested changes are present and recognizable, but with small imperfections—e.g., slightly off colors, minor typos (”Summmer” instead of ”Summer”), or incorrect font/texture.

5: Perfect execution: The instruction is executed exactly as requested with no errors or ambiguities.

IMPORTANT:
Evaluate ONLY instruction execution here. Ignore aesthetics unless they prevent instruction execution.

CRITERION 2: IDENTITY PRESERVATION

Question:
Did the edit preserve all parts of the original image that were NOT explicitly requested to change?

Scoring (0–5):

0: Catastrophic over-editing: The edit has completely destroyed the original layout and composition of the image, the new image does not preserve anything from the original image.

1: Significant over-editing: The edited image retains some parts of the original image, but the new layout is still extremely different, such as unnecessary zooming into the product or removing the background.

2: Moderate over-editing: The layout is mostly preserved, but few elements such as important objects, or critically important informative text boxes have been changed unnecessarily.

3: Minor over-editing: Small, yet noticeable changes, such as removing less important object/label or adding a new one. Performing add instead of modify etc.

4: Negligible over-editing: Subtle artifacts, such as different outline color, or minor color/texture changes.

5: Perfect preservation: Nothing has been unnecessarily changed.

CRITERION 3: VISUAL QUALITY

Question:
Does this image have good visual quality and aesthetics for use as an advertisement?

Scoring (0–5):

0: Aesthetically broken: All regions of the image are visually unpleasant or chaotic (e.g., severe artifacts, incoherent composition, unreadable text, clashing elements). It does not function as a usable advertisement.

1: Poor aesthetics: Not all parts, but majority of the image has major quality issues—heavy blurring, over-saturation, or distortion that makes it look unprofessional.

2: Weak aesthetics: Important regions suffer from noticeable quality issues—blurring, over-saturation, warping, or cluttered placement/overlapping text regions—that make the ad look amateurish. Otherwise the image looks acceptable.

3: Acceptable aesthetics: The image is usable but has style issues—low-quality textures, fonts that look out of place, layout is not nice on the eye etc.

4: Good aesthetics: Visually pleasing and well-composed overall, with only minor aesthetic imperfections, such as text which is slightly garbled or objects which slightly overlap.

5: Excellent aesthetics: Highly polished, visually harmonious, and professionally composed. All elements work together cleanly and attractively as a high-quality advertisement. None of the components objectively detract from the overall visual appeal.

To compute these scores, we use GPT-5 as a strong judge to evaluate the edited images according to the defined criteria. Finally, we aggregate the three criterion scores into a single scalar reward. Let IE, IP, and VQ denote the scores for Instruction Execution, Identity Preservation, and Visual Quality respectively. Rather than summing the scores, which would allow a high score in one dimension to compensate for poor performance in another, we compute the geometric mean of the three scores:

$$R = (\text{IE} \cdot \text{IP} \cdot \text{VQ})^{1/3}.$$

This formulation encourages balanced performance across all criteria, since a low score in any single dimension significantly reduces the overall reward. Furthermore, many of the editing tools used in our pipeline (e.g., diffusion-based models) are inherently stochastic and may produce different outputs for the same input due to sampling noise. To reduce the variance, we generate two outputs for each tool invocation. The reward for a given tool selection is then computed as the average reward across these two outputs. Now we have a dataset where we have precomputed the tool output for every instruction given the original image on the training set and we have scored them, therefore we know which tools have the highest reward and can train on them.

Reward Model for Inference During inference, verification is critical to avoid selecting a poor editing action that could negatively affect subsequent steps in the editing process. Since the or-

chestrator considers multiple candidate tool executions, we require a reward model to evaluate the resulting edits and select the most desirable outcome.

Although the rubric described above can be evaluated using a strong closed-source judge, relying on such models during inference would be computationally expensive. To keep inference costs manageable, we instead distill this evaluation signal into a lightweight open-source reward model.

Specifically, we use *Qwen3-VL-8B* as the backbone and train separate classification heads to predict the evaluation scores for each criterion. Each head predicts the score distribution for one of the three axes—Instruction Execution (IE), Identity Preservation (IP), and Visual Quality (VQ)—using supervision derived from the judge's scores for each tool execution. This allows the model to approximate the behavior of the original evaluator while remaining efficient enough for use during inference.

During inference, the reward model outputs logits over the possible score levels for each criterion. These logits are converted into probabilities using a softmax, and the expected score for each criterion is computed by weighting the possible score values by their predicted probabilities. Formally, if $z_{c,k}$ denotes the logit corresponding to score level $k \in \{1, \ldots, 5\}$ for criterion $c \in \{\text{IE}, \text{IP}, \text{VQ}\}$, the expected score is computed as

$$\hat{s}_c = \sum_{k=0}^{5} k \cdot \text{softmax}(z_c)_k.$$

The final reward is then obtained by aggregating the predicted scores using the same geometric mean formulation used during training:

$$R = (\hat{s}_{\text{IE}} \cdot \hat{s}_{\text{IP}} \cdot \hat{s}_{\text{VQ}})^{1/3}.$$

C.3 Evaluation

Open-ended image editing, and in particular advertisement editing, has not been studied in detail by prior work. Therefore, we need to design a comprehensive measure of success. In the paper, we report results in the main section 4.1 of experiments as well as the ablations in section 4.2. In addition, we also compare the two plans in Section B.2. In this section, we provide the details of each of these evaluations.

C.3.1 Whole-Edit Evaluation

We aim to evaluate both the quality of the generated plan and the correctness of its execution by assessing the final edited image. To do so, we require a metric that captures both the reasoning and knowledge demonstrated by the planner when determining the required modifications, as well as the system's ability to faithfully execute those modifications and produce a high-quality image that remains semantically consistent with the user's request.

To this end, we evaluate edits along three axes: *Instruction Execution*, *Identity Preservation*, and *Visual Quality*. The judgement is conditioned on the initial image, the final edited image, and the high-level task description. Since we trained our model using GPT based rewards, in order to remove any potential bias in evaluation, we use Gemini-3-Pro (Google DeepMind, 2026) as a judge here.

For *Instruction Execution* we use the following rubric to score the edits:

Instruction Execution (IE)
System Instruction: You are an expert evaluator for image editing quality with a focus on *instruction execution*. Your task is to determine how accurately the requested edit has been carried out in the edited image.
Evaluation Description: The model must do two things: (1) identify what should change in the original image to achieve the requested theme, and (2) correctly execute those changes. Scoring considers *intent* (correct understanding of what must change), *coverage* (whether sufficient changes are made to convey the theme), and *execution* (whether those changes are implemented correctly). Compare the original image with the edited image and determine whether the intended transformation has been properly realized.
Scoring Guide
1 — Very Poor: Intent is incorrect. The instruction is misunderstood and the requested transformation is not meaningfully reflected (e.g., unrelated edits or changing the subject).
2 — Below Average: Intent is roughly correct but execution is poor, contradictory, or incorrect. Alternatively, execution may be reasonable but the edits are too minimal to convey the requested theme.
3 — Acceptable: Intent and coverage are reasonable, but execution contains several flaws. Multiple sub-edits may be inconsistent or visually incorrect.
4 — Good: Intent and coverage are good and execution is mostly correct. Minor imperfections may exist, but the overall transformation clearly reflects the instruction.
5 — Excellent: Intent is precise, coverage is complete, and execution is correct throughout. All necessary edits are applied and the resulting image fully conveys the requested transformation.

With this rubric, we observe that the judge model evaluates the edit based on both the knowledge demonstrated as well as the success of execution.

In order to measure *Identity Preservation*, we use the following rubric:

Identity Preservation (IP)
System Instruction: You are an expert evaluator for image editing quality with a focus on *identity preservation*. Your task is to evaluate how well the edited image preserves the core identity and non-target content of the input image.
Evaluation Description: Identity preservation measures how well the edited image maintains the core identity and non-target content from the input image while allowing necessary attribute changes required by the editing instruction. Acceptable modifications may include changes to color, texture, background, style, or other properties when they are explicitly required to follow the instruction. Changes that go beyond what is necessary, introduce unintended drift, or alter unrelated elements of the image should reduce the score.
Scoring Guide
1 — Very Poor: Core identity is heavily altered or unrecognizable compared to the input image. Major non-target content is changed without justification. Extensive unintended modifications or over-editing are present.
2 — Below Average: Significant identity drift beyond what is required by the instruction. Multiple unnecessary changes to attributes, structure, or background. Edits exceed what is needed to satisfy the instruction.
3 — Acceptable: Core identity is mostly preserved and required attribute changes are applied, but there is noticeable drift in some structural elements or non-target regions beyond what the instruction necessitates.
4 — Good: Core identity and non-target content are well preserved. Attribute changes (e.g., color, texture, background) are applied only when justified by the instruction. Minor unintended drift or slight over-editing may be present.
5 — Excellent: Core identity is faithfully preserved relative to the input image. Only the modifications necessary to fulfill the instruction are applied, and unrelated content remains unchanged.

And for visual quality, we only take in the final image as input (it is independent of either the instruction or the initial image) and evaluate the quality based on the following rubric:

Visual Quality (VQ)
System Instruction: You are an expert evaluator for image editing quality with a focus on *visual quality*. Your task is to evaluate the overall visual quality, coherence, and artifact level of the edited image.
Evaluation Description: Visual quality measures how clearly and professionally the edited image presents information while remaining free from artifacts, distortions, or rendering errors. The image should appear polished and visually coherent, with edited regions integrated naturally with the rest of the image. Quality issues such as blurriness, broken structure, layout inconsistencies, or rendering artifacts should reduce the score.
Scoring Guide
1 — Very Poor: Severe artifacts, distortions, broken anatomy or structure, or major rendering failures. Strong blurriness. Text is completely garbled, blurred, or unreadable and conveys no information. Layout is unusable and information is effectively lost due to quality issues.
2 — Below Average: Obvious artifacts (e.g., warped regions, visible overlaps) affecting important objects or regions. Text has issues in every block and important information is not clearly conveyed. Typos may obscure meaning. Major objects may overlap incorrectly. Visual flaws significantly reduce usability.
3 — Acceptable: Generally usable image quality, but noticeable issues such as mild artifacts, slight blur, or weak integration between edited and original regions. Textures and text are mostly correct but may show minor distortions or typos. Color, font, or layout consistency may be poor. The image appears amateurish but information remains understandable.
4 — Good: Clean and coherent image with minor imperfections. Textures and edited regions blend well with the original content. Text is readable and correctly placed but may have minor stylistic issues (e.g., color inconsistencies or symmetry problems). Some imperfections in lighting, perspective, or design choices may remain.
5 — Excellent: High-fidelity, sharp, and visually coherent image. Lighting and shadows are natural and consistent. No visible artifacts, distortions, blur, or rendering errors. Edited content is seamlessly integrated with the original image. Text is crisp, readable, and free of overlap or distortion.

This allows us to compare methods under a common high-level instruction while different techniques attempt to solve the task using different strategies.

C.3.2 PLAN-CONDITIONED EVALUATION

In the ablation studies presented in the main paper, we compared several components of our method while keeping the underlying plan fixed. The goal of this experiment is to determine whether, given the same plan, the combination of editing tools and a learned orchestration policy leads to improved performance.

Under this setting, evaluating instruction execution and identity preservation becomes more specific. In addition to assessing whether the overall task is addressed, we must also evaluate whether the resulting edits follow the plan itself. To enable this, we rely on a dense checklist derived from the plan.

Concretely, given the input image, the high-level instruction, and the multi-step plan, we use a strong MLLM to generate a dense checklist describing the criteria that the edited image should satisfy. The checklist enumerates specific concepts that should be modified, preserved, added, or removed, as well as important relationships between objects in the scene that should remain consistent with the plan.

During evaluation, we again use Gemini-3-Pro (Google DeepMind, 2026). The judge receives the original image, the high-level task, the edited image, and the generated checklist, and determines whether each checklist item is satisfied or not. The final score for an image is computed as the fraction of checklist items that are satisfied. We then average these scores across the dataset to obtain the reported results.

Checklist Generation In order to generate the checklist, we use the following system prompt:

Checklist Specification Generator Prompt
System Instruction: You are a specification generator for image editing tasks. Your goal is to produce a *final-state constraint list* describing what must be true after all edits are completed.
Inputs:

- Input image
- Task description
- Step-by-step editing plan

Output Format: Return a valid Python list of strings. Each string must represent exactly **one atomic final-state constraint**. Use the following prefixes:

- **Preserve:**
- **Remove:**
- **Replace:**
- **Add:**
- **Constraint:**

Rules:

- Analyze the input image to identify salient and structurally important elements.
- If an important element is **not modified by the plan**, include a **Preserve** constraint.
- For replacements, use the format: `Replace: <original> -> <new>`.
- Use **Remove** only when an element must explicitly disappear.
- Use **Add** only for new elements introduced by the plan (include placement or attributes when relevant).
- Use **Constraint** only when the plan specifies positional, visibility, or relational requirements (e.g., top-right corner, left of product, below logo).
- Do **not** include aesthetic or realism requirements unless explicitly stated in the plan.
- If multiple valid realizations exist, include all possible outcomes.
- Each list entry must represent exactly **one atomic execution condition**.
- Be concise.
- No explanations.
- No markdown.
- Output only the Python list.

Example
Image contains:

- Product pack
- Brand logo
- Blue background
- Wooden surface

Plan:

1. Replace background with a festive Diwali scene
2. Add fireworks

Output:

```
[
  "Preserve: product pack design",
  "Preserve: brand logo",
  "Preserve: wooden surface",
  "Replace: background -> Diwali festive scene",
  "Add: fireworks"
]
```

This prompt helps us to generate a dense checklist. Now we use this dense checklist to score the final edit. The system prompt for that is:

Checklist Verification Prompt
System Instruction: You are a strict evaluator for image editing execution. Your task is to determine whether the edited image satisfies each constraint in a provided checklist.
Inputs:

- Original (input) image
- Editing task / instruction
- Checklist of final-state constraints
- Edited (output) image

Checklist Format: Each checklist item begins with one of the following prefixes:

- **Preserve:**
- **Remove:**
- **Replace:**
- **Add:**
- **Constraint:**

Evaluation Rules:

- **Preserve:** Mark Y if the referenced element remains present and meaningfully unchanged. Mark N if it is altered, removed, or obscured.
- **Remove:** Mark Y if the element is no longer visible. Mark N if it still appears.
- **Replace:** Mark Y only if the original element is gone and the specified new element is present. Otherwise mark N.
- **Add:** Mark Y if the specified new element clearly exists. Mark N if it is missing or incorrect.
- **Constraint:** Mark Y if the stated positional, relational, or visibility condition is satisfied. Otherwise mark N.

General Guidelines:

- Judge strictly based on visible evidence in the edited image.
- Do not assume intent or infer missing details.
- If the result is unclear or ambiguous, mark N.
- Ignore aesthetic quality unless it is explicitly part of the constraint.
- Provide only a very short justification phrase (e.g., "logo still visible", "background changed", "color incorrect").

Output Format: Return a JSON array where each checklist item corresponds to one entry containing a short reason and a binary result.

```
[
  { "reason": "<short reason>", "result": "Y" },
  { "reason": "<short reason>", "result": "N" },
  ...
]
```

Output only the JSON array and no additional text.

For visual quality, we use the same evaluation as we discussed in section C.3.1.

C.3.3 PLAN COMPARISON

In order to perform the evaluation in section B.2. We use the following prompt:

Plan Preference Evaluation Prompt
System Instruction: You are an expert evaluator of image editing plans. Your task is to compare two candidate plans and determine which one better solves the given editing task.
Inputs:

- Input image
- Editing task / instruction
- Plan A
- Plan B

Evaluation Criteria: Choose the plan that demonstrates stronger understanding of the image and the editing task.
Strong plans typically:

- correctly identify the important elements in the image that must be modified
- recognize which elements should be preserved
- demonstrate practical knowledge of how edits would realistically be executed
- propose non-trivial yet practical changes that lead to a meaningful edit
- provide clear and direct instructions with minimal ambiguity

Weak plans often:

- give generic instructions without understanding the structure of the image
- rely on vague wording indicating limited knowledge of the task
- suggest trivial changes that do not meaningfully improve the image
- contain vague steps (e.g., "update the text" without specifying how)

Plan Consistency Rules:

- Plans are sequential; later steps operate on the result of earlier steps.
- Early steps should not remove or modify elements in a way that makes later steps impossible.
- Intermediate steps need not make sense in isolation, only within the overall plan.

Output Format: First briefly explain your reasoning (2–4 sentences) referencing specific steps or phrases from the plans. Then output the final choice as JSON.

```
{
  "reasoning": "<brief reasoning>",
  "choice": "A" or "B"
}
```

C.4 Inference through Tree Search

Each tool invocation is represented using a structured format

$$\{\texttt{"tool"}: t,\ \texttt{"arguments"}: a\},$$

where t denotes the tool name and a specifies the corresponding arguments (e.g., region selection). This representation mirrors the execution interface of our editing framework, allowing predicted tool calls to be directly executed without requiring the model to generate intermediate code.

Importantly, this structured representation also defines a discrete and enumerable space of candidate actions. In typical tool-calling setups where the model generates free-form code or API calls, the output space is effectively unbounded, making it difficult to evaluate likelihoods over all possible actions. In contrast, our formulation specifies a finite set of candidate tool–region pairs $(a, r) \in \mathcal{C}$ for each sub-task. This allows us to explicitly evaluate how likely the orchestrator considers each candidate action.

Let $y_{a,r} = (y_1, \ldots, y_L)$ denote the token sequence corresponding to a candidate tool invocation. For each candidate action we compute a length-normalized log-likelihood score under the orchestrator policy π_ϕ:

$$\text{score}_{a,r} = \frac{1}{L} \sum_{i=1}^{L} \log \pi_\phi(y_i \mid \hat{x}, s_t, y_{<i}).$$

This quantity corresponds to the average token log-likelihood (equivalently, the negative log-perplexity up to a constant) assigned by the orchestrator to the candidate action. Because the candidate action space is explicitly enumerated, the orchestrator can score and rank all possible tool selections in this manner.

These scores are then used to select the most promising candidate actions, which are executed and evaluated by the reward model as described in Algorithm 1.

Algorithm 1: Reward-Guided Tool Selection

Input: Input image x, plan $\mathcal{P} = \{s_1, \ldots, s_T\}$, orchestrator π_ϕ, reward model R, candidate tool–region set $\mathcal{C}$, beam size K

Output: Edited image $\hat{x}$

$\hat{x} \leftarrow x$

for *each sub-task* $s_t \in \mathcal{P}$ **do**

  for *each candidate* $(a, r) \in \mathcal{C}$ **do**

    Let $y_{a,r} = (y_1, \ldots, y_L)$ be the token sequence for (a, r);

    $\text{score}_{a,r} \leftarrow \frac{1}{L} \sum_{i=1}^{L} \log \pi_\phi(y_i \mid \hat{x}, s_t, y_{<i})$;

  $\mathcal{B}_t \leftarrow \text{TopK}_{(a,r)\in\mathcal{C}}(\text{score}_{a,r}, K)$;

  for *each* $(a, r) \in \mathcal{B}_t$ **do**

    $\tilde{x}_{a,r} \leftarrow f_{a,r}(\hat{x})$;

    $R_{a,r} \leftarrow R(\tilde{x}_{a,r}, x)$;

  $(a_t, r_t) \leftarrow \arg\max_{(a,r)\in\mathcal{B}_t} R_{a,r}$;

  $\hat{x} \leftarrow f_{a_t,r_t}(\hat{x})$;

return $\hat{x}$