CogniEdit: Dense Gradient Flow Optimization for Fine-Grained Image Editing

Supplementary Material

7. Algorithm

We provide the training procedure for CogniEdit, which combines Dynamic Token Focus Relocation with Dense GRPO Optimization.

Algorithm 1 Training Procedure of CogniEdit

Require: Policy π_θ, reference π_{ref}, reward model R, MLLM, batch size B, group size G, timesteps T, dense steps k

Initialize θ, token focus predictor p_η, soft tokens $\{s_i^{1:k}\}_{i=1}^N$
for each training iteration **do**
  Sample batch $\{(\text{image}_b, \text{instruction}_b)\}_{b=1}^B$
  Decompose instructions: $c_b \leftarrow \text{MLLM.decompose}(\text{instruction}_b)$ for $b = 1, \ldots, B$
  Dynamic Token Focus: For each layer i, predict pos $\leftarrow p_\eta(h_i^l)$ and inject $h_i^{\text{pos:pos}+k} \leftarrow s_i^{1:k}$
  for $b = 1$ to B **do**
    Sample $r \sim \text{Uniform}[k, T]$ and initialize $x_r^{b,1:G}$
    for $i = 1$ to G **do**
      Denoise k steps: $x_{t-1}^{b,i} \leftarrow x_t^{b,i} - \frac{1}{T}(v_\theta(\text{sg}(x_t^{b,i}), t) + \frac{\sigma_t^2}{2}[x_t^{b,i} + (1-t)v_\theta(\text{sg}(x_t^{b,i}), t)]) + \frac{\sigma_t}{\sqrt{T}}\epsilon$ for $t = r, \ldots, r-k$
      Complete: $x_0^{b,i} \leftarrow \text{sg}(\text{Denoise}(x_{r-k}^{b,i}))$
      Evaluate: $R(x_0^{b,i}, c_b)$
    end for
  end for
  Compute the batch advantages: $\hat{A}^b \leftarrow \frac{1}{G}\sum_{i=1}^G \frac{R(x_0^{b,i}, c_b) - \mu_{\text{batch}}}{\sigma_{\text{batch}}}$ where $\mu_{\text{batch}}, \sigma_{\text{batch}}$ are batch statistics
  Compute probability ratios: $\tilde{r}_b^{r:r-k}(\theta) \leftarrow \exp(\text{clip}(\sum_{t=r}^{r-k+1} \log \frac{p_\theta(x_{t-1}^b | x_t^b, c_b)}{p_{\theta_{\text{old}}}(x_{t-1}^b | x_t^b, c_b)}, -\log(1+\epsilon), \log(1+\epsilon)))$
  Compute optimization objective: $\mathcal{J}_{\text{Dense}}(\theta) \leftarrow \frac{1}{B}\sum_{b=1}^B \min(\tilde{r}_b^{r:r-k}\hat{A}^b, \text{clip}(\tilde{r}_b^{r:r-k}, 1-\epsilon, 1+\epsilon)\hat{A}^b) - \hat{\beta} D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$
  Update: $\theta, \eta, \{s_i^{1:k}\}_{i=1}^N \leftarrow$ gradient step on maximizing $\mathcal{J}_{\text{Dense}}(\theta)$
end for
return $\pi_\theta, p_\eta, \{s_i^{1:k}\}_{i=1}^N$

The algorithm integrates three key components: (1) MLLM-based instruction decomposition for semantic understanding, (2) Dynamic Token Focus Relocation for hierarchical attention patterns, and (3) Dense GRPO Optimization with gradient accumulation across k consecutive denoising steps. The batch-level advantage computation stabilizes training for editing tasks, while stop gradient operations ensure proper gradient flow through the sampling trajectory.

8. Background

8.1. Stop Gradient

Stop gradient is a technique that selectively blocks gradient flow through specific computational paths during backpropagation. In neural network training, gradients typically flow backward through all operations to update model parameters. However, in certain scenarios, allowing unrestricted gradient flow can lead to training instability or undesired parameter updates. By applying stop gradient operations, denoted as $\text{sg}(\cdot)$, we can treat certain values as constants during the backward pass while preserving their computational graph during the forward pass.

Formally, for a tensor x, the stop gradient operation is defined as:

$$y = \text{sg}(f(x)), \quad \text{where} \quad \frac{\partial y}{\partial x} = 0 \tag{10}$$

During forward propagation, $y = f(x)$, but during backpropagation, no gradients flow through this operation. This is particularly useful in reinforcement learning and self-supervised learning scenarios where we want to compute values based on the current model state without updating certain components.

In our framework, we strategically employ stop gradient operations to stabilize training when computing target values or baseline estimates that should not directly influence certain model components through gradient descent, as we detail in Section 3.3.

9. Methodology

9.1. Visual Reasoning Enhanced Text Encoding

Visual reasoning enhanced text encoding leverages vision-language models (VLMs) to generate enriched text representations that exhibit superior alignment with visual content. Recent research has demonstrated that exploiting the reasoning capabilities of VLMs to extract detailed, fine-grained visual information yields more effective guidance for model behavior [34, 43]. This approach addresses the limitation of naive text encoding, which often fails to capture the nuanced semantic relationships between instructions and visual content necessary for precise editing operations.


Input
Ours
Qwen-Image
OmniGen2
Step1X-Edit
"Correct the Incorrect Driving Habits in the Image"
"Place Christian classic symbol items on the table"
"Change to resemble the appearance after being exposed under the sun for half an hour."
"Add some snow to the background"
"Apply a suitable filter for this image"
"Make the ice cream jade"

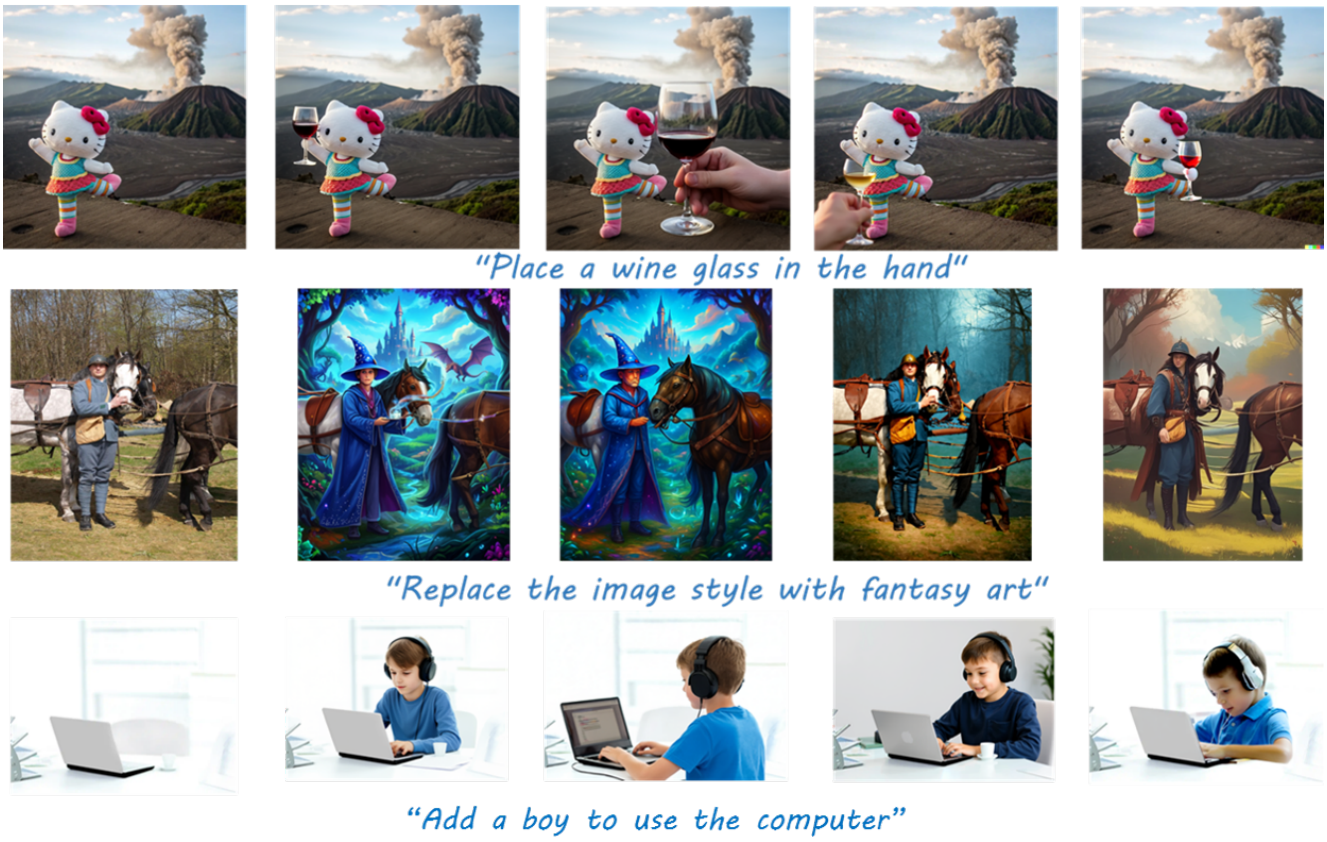


Figure 9. Qualitative comparison with instruction-based editing methods. CogniEdit maintains high visual quality and natural blending between edited and unedited regions, while baseline methods struggle with consistency preservation.

The encoding process follows a structured reasoning pipeline. Given an editing instruction and source image, the VLM performs hierarchical semantic analysis through three stages: (1) **Visual Comprehension** – generating a detailed image description encompassing objects, quantities, textual elements, spatial relationships, and salient visual features; (2) **Reasoning** – constructing a logical reasoning chain that connects the visual content with the editing instruction; (3) **Instruction Synthesis** – producing an enriched editing instruction that incorporates both the original command and contextual visual information.

To further enhance editing precision, we extend this standard reasoning framework with an additional **Target Specification** stage that explicitly identifies and describes the specific image regions or objects to be modified. This four-stage reasoning process provides comprehensive semantic grounding for the editing model, enabling more accurate interpretation of editing intents.

Formally, we prompt the VLM to generate structured outputs following the format:

- **Caption** (`<info>`): A comprehensive visual description capturing objects, quantities, text, spatial relations, and salient features;
- **Reasoning** (`<think>`): A logical reasoning chain connecting visual content with the editing instruction;
- **Answer** (`<answer>`): An enriched editing instruction synthesizing original command and visual context;
- **Target** (`<object>`): Detailed specification of regions or objects to be modified.

The complete output structure is thus:

```
<info>...</info> <think>...</think>
<answer>...</answer> <object>...</object>
```
(11)

,which provides the editing model with multi-level semantic guidance ranging from holistic scene understanding to precise target localization.

9.2. Data Preparation

We construct our training dataset by combining existing editing datasets with self-constructed data, resulting in a diverse collection of image editing pairs. Specifically, we utilize the following data sources:

Figure 10. Qualitative comparison with knowledge-based editing methods. Each example shows source image, instruction requiring domain knowledge or reasoning, and results from different methods.

- **SEED-Data-Edit [10]** [3]: We sample 3k editing pairs from Part 2 of this dataset, which comprises real-world editing scenarios collected from the internet.
- **COCO 2017 [22]** [4]: We randomly select 1k images from the COCO 2017 dataset and construct corresponding editing pairs.

[3] https://huggingface.co/datasets/AILab-CVC/SEED-Data-Edit

[4] https://cocodataset.org/

For the self-constructed COCO-based data, we employ a systematic approach to generate editing pairs. For each selected image, we apply instance segmentation to obtain object masks and select the largest segment as the target editing region based on mask area. We then create editing instructions following the template: "Add a green segmentation mask (for object detection) to the *[object name]* on the *[position]* of the image", where *[object name]* and *[position]* are automatically determined from the segmentation results. This construction process yields paired data consisting of the original image, the edited image with the applied mask, and the corresponding text instruction.

10. Knowledge-enhanced instruction

Complex editing instructions often contain implicit semantic requirements or require domain knowledge that is challenging for editing models to directly interpret. For instance, an instruction like "Replace the sky with a sunset scene" requires understanding what constitutes a visually plausible sunset (warm color gradients, sun position, atmospheric effects), while "Change the car to a sports car" demands knowledge of typical sports car characteristics (low profile, aerodynamic design, distinctive features). To bridge this semantic gap, we leverage Multi-modal Large Language Models (MLLMs) to decompose and enhance editing instructions with explicit, actionable guidance.

MLLM-based Instruction Enhancement. Given a source image and an editing instruction, we employ an MLLM (specifically, PeBR-R1-7b [6]) to analyze the visual content and generate a knowledge-enhanced instruction. The enhancement process follows a structured prompt template that guides the MLLM to:

- **Semantic Decomposition**: Break down abstract editing requests into specific visual attributes (colors, shapes, positions, textures).
- **Knowledge Injection**: Incorporate domain-specific knowledge relevant to the editing task (e.g., anatomical correctness, physical plausibility).
- **Spatial Grounding**: Provide explicit spatial references to guide precise localization of editing regions.

The used template is presented in Figure 12. The MLLM processes both the image and instruction, generating enhanced instructions that explicitly specify editing details. And we provide samples of the enhanced instructions in Figure 13, including the description of the input image, the think process and the enhanced instruction. The knowledge enhancement provides several key advantages. First, it transforms ambiguous instructions into explicit specifications, reducing the model's burden of implicit reasoning during the editing process. Second, it ensures semantic consistency by incorporating factual knowledge (e.g., correct number of petals for specific flowers, appropriate proportions for objects). Third, it provides fine-grained attribute specifications that align with our Dynamic Token Focus Relocation mechanism—the enhanced instructions contain more tokens describing precise attributes, which our mechanism can effectively emphasize during processing.

11. Experiment

11.1. Reward Model

Effective reward signals are critical for GRPO optimization. We design a comprehensive reward model based on Multimodal Large Language Models (MLLMs) that evaluates edited images across multiple dimensions. Unlike single-metric rewards (e.g., CLIP score) that only measure text-image alignment, our reward model provides multi-faceted evaluation considering instruction following, visual coherence, and source consistency—essential criteria for high-quality image editing.

MLLM-based Reward Evaluation. We employ Qwen2.5-VL-7B-Instruct as our reward model. Given an edited image, the original image, and the editing instruction, the model evaluates the editing quality through a structured prompt that assesses multiple aspects:

1. **Word-level Analysis**: Extract key words related to subjects, objects, colors, numbers, lighting, style, and activities in the instruction, scoring how well each element is visually represented in the edited image. A special token `[No_mistakes]` indicates whether all elements are correctly depicted.
2. **Holistic Assessment**: Provide overall scores on four axes (each rated 1-5):
   - *Alignment Score*: How well the edited image matches the instruction in terms of content.
   - *Coherence Score*: Logical consistency of the image, including absence of visual glitches, object distortions, location errors, or incorrect colors/quantities.
   - *Style Score*: Aesthetic quality and visual appeal of the edited image.
   - *Consistency Score*: Fidelity to the original image in unedited regions.

   The complete reward prompt is:

   > *You are presented with an edited image (the first image) and its original image (the second image), and the associated text caption of the edited image. Your task is to analyze the edited image across multiple dimensions in relation to the caption and its original image. Specifically:*
   >
   > *1. Extract key words related to: subject, object, color, number, lighting, style and activities in the caption based on how well it is visually represented in the edited image. - A higher score indicates that the word is less well represented in the image. - The special token [No_mistakes] represents whether all elements in the caption were correctly depicted. A high score suggests no mistakes; a low score suggests missing or incorrect*

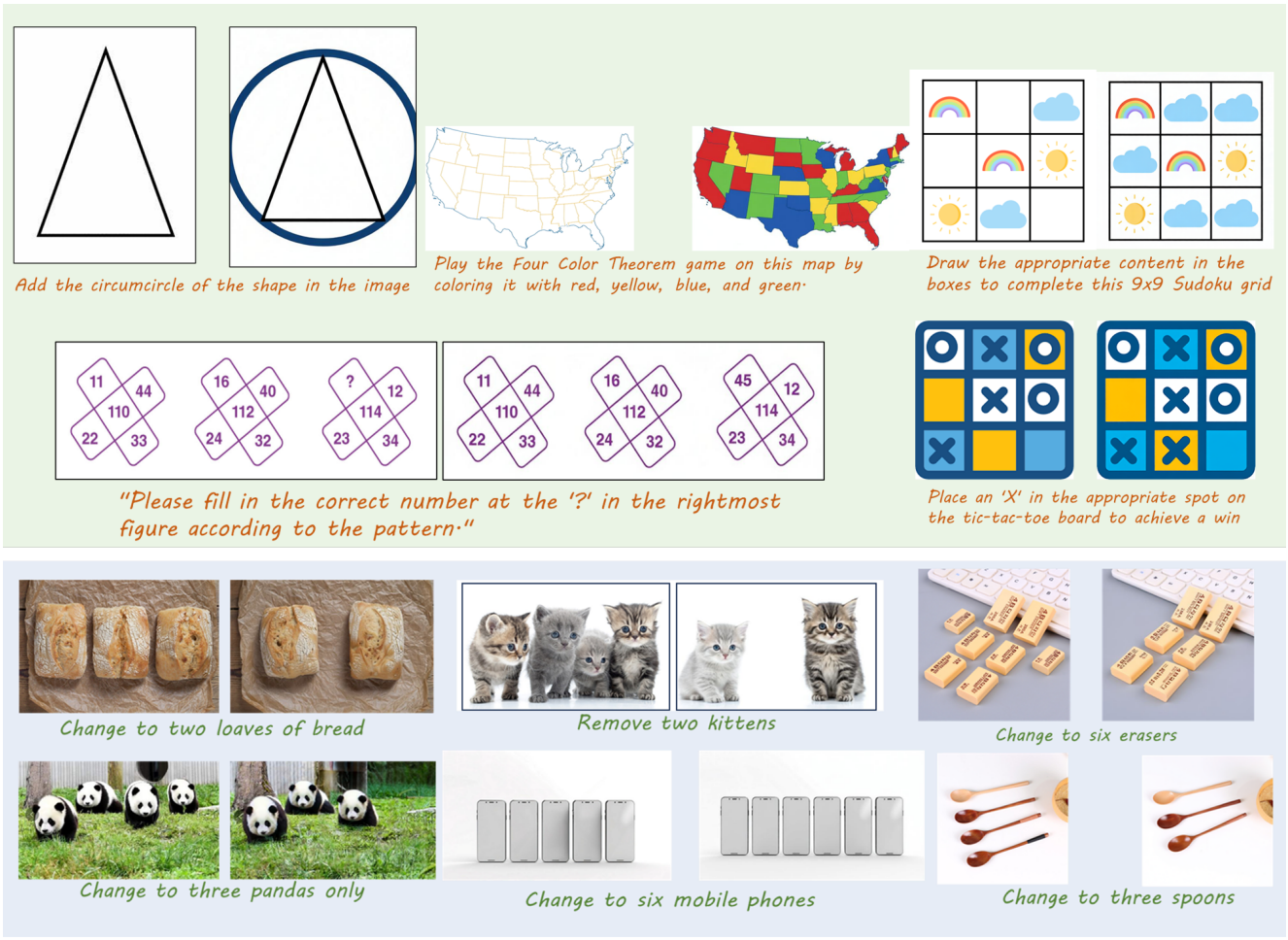


Figure 11. The editing results of our method on Kris-Bench.

elements. 2. Provide overall assessments for the image along the following axes (each rated from 1 to 5): - Alignment Score: How well the image matches the caption in terms of content. - Coherence Score: How logically consistent the image is (object location error, absence of visual glitches, object distortions, object is missing or extra, color or quantity error etc.). - Style Score: How aesthetically appealing the image looks, regardless of caption accuracy. - Consistency Score: How well the edited image consistent to its original image.

Output your evaluation using the format below and also provide the reason why you give this score: Alignment Score (1-5): X Coherence Score (1-5): Y Style Score (1-5): Z Consistency Score (1-5): W Output the basis and reasons for the scores given above.

Reward Computation. From the MLLM's structured output, we extract three key metrics to compute the final reward. We exclude the Style Score as it measures aesthetic appeal independent of editing correctness. The reward R for each edited image is computed as:

$$R = S_{\text{alignment}} + S_{\text{coherence}} + S_{\text{consistency}}, \tag{12}$$

where $S_{\text{alignment}}$, $S_{\text{coherence}}$, and $S_{\text{consistency}}$ are the alignment, coherence, and consistency scores respectively, each normalized to [0, 5]. This formulation balances three critical aspects: (1) instruction-image alignment ensures the edit follows the textual specification, (2) coherence prevents visual artifacts and semantic inconsistencies, and (3) consistency preserves the integrity of unedited regions. By averaging these three scores, we obtain a holistic reward that guides the model to produce edits that are simultaneously accurate, plausible, and faithful to the source image. We provide the samples of how the reward model evaluates the edited image in Figure 14.

Advantages of MLLM-based Rewards. Compared to traditional metrics like CLIP score or learned discriminators, our MLLM-based reward model offers several advantages. First, it provides interpretable, fine-grained feedback through word-level analysis and multi-dimensional scoring.

Prompt: You are tasked with analyzing an image to generate an exhaustive and detailed description. Your goal is to extract and describe all possible information from the image, including but not limited to objects, numbers, text, and the relationships between these elements. The description should be as fine and detailed as possible, capturing every nuance. After generating the detailed description, you need to analyze the instruction and provide step-by-step simple reasoning for the given instruction, please avoid repeating the same points. Finally, provide concise answer to what will happen with the given instruction and provide the specific instruction to modify the image, and list the properties of the modified image, including number, color and position of the object. The description, reasoning process, answer and property are enclosed within <info> </info>, <think> </think>, <answer> </answer> and <object> </object> tags, respectively, i.e., <info> image description here </info> <think> reasoning process here </think> <answer> answer here </answer> <object> list the perperties of modified image here </object>.

Figure 12. The template used for instruction enhancement.

Second, it can evaluate complex semantic correctness (e.g., correct number of objects, appropriate spatial relationships) beyond simple feature matching. Third, by conditioning on both the source and edited images, it explicitly evaluates source consistency—a crucial aspect often overlooked by generation-focused metrics. Empirically, we find that this reward formulation leads to more stable training and better alignment with human preferences compared to single-metric alternatives.

11.2. Results on GEdit-Bench

Editability Preservation. Our dense reward optimization is specifically designed to enhance fine-grained alignment with detailed prompt features (e.g., precise colors, positions, quantities), which represents a different optimization objective than general-purpose editing. To assess whether this specialization compromises broader editing capabilities, we evaluate on GEdit-Bench with three metrics: Q_SC (source consistency), Q_PQ (edited quality), and Q_O (overall quality), evaluated by QwenVL-2.5-72B. As shown in Table 3, our method maintains competitive performance across both the intersection subset and full set, with scores comparable to top-performing general-purpose models like Qwen-Image and Step1X-Edit. While slight variations in individual metrics reflect the inherent trade-off between optimizing for fine-grained precision versus general editing flexibility, our method does not exhibit significant degradation despite being optimized for a more specialized objective. These results validate our design choice: we achieve substantial improvements on fine-grained tasks (as shown in Kris-Bench) without sacrificing competitiveness on general editing scenarios, demonstrating that dense reward optimization successfully balances fine-grained instruction following with general editability. The editing results of our method on the GEdit-Bench are shown in Figure 15.

Table 3. The results on the GEdit-Bench, where Q_SC is the quality score of the source image, Q_PQ is the quality score of the edited image, and Q_O is the quality score of the edited image..

Method	Intersection subset			Full set		
	Q_SC	Q_PQ	Q_O	Q_SC	Q_PQ	Q_O
InsPix2Pix	4.833	6.992	4.691	4.746	6.913	4.578
MagicBrush	5.814	7.149	5.653	5.752	7.069	5.558
AnyEdit	3.873	6.754	3.789	3.713	6.730	3.635
Step1X-Edit	7.501	7.264	7.189	7.388	7.279	7.067
OmniGen2	6.584	7.233	6.295	6.618	7.191	6.296
Qwen-Image-Edit	**7.819**	7.398	**7.462**	**7.752**	7.394	**7.402**
Ours	7.723	**7.441**	7.408	7.618	**7.417**	7.285

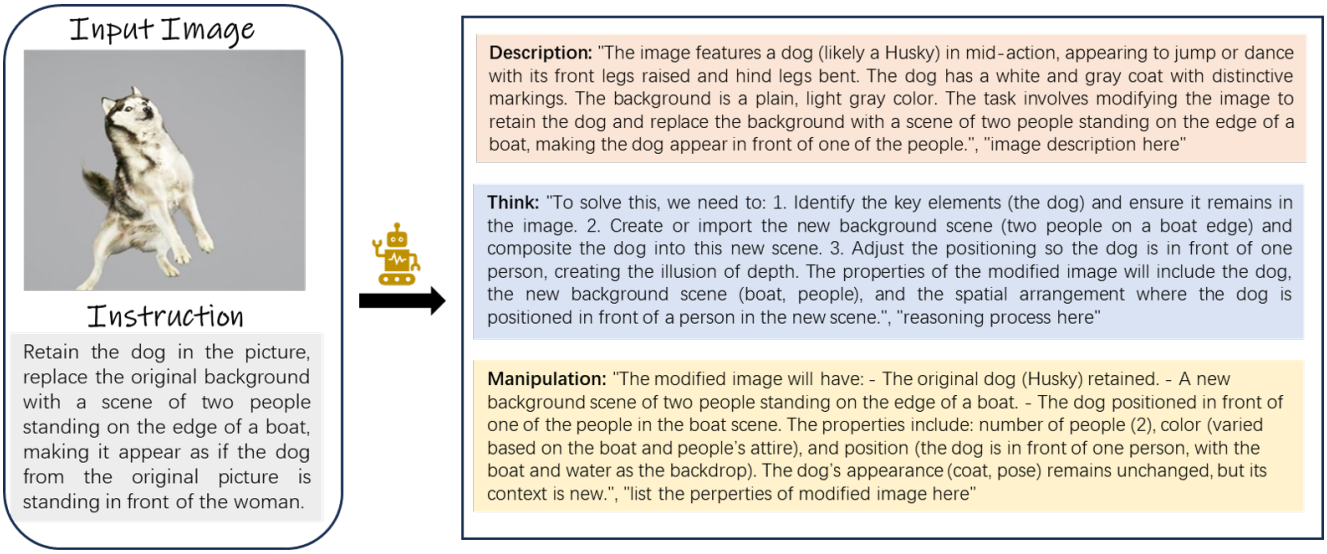
11.3. Visualization

We provide qualitative comparisons to demonstrate CogniEdit's superiority in handling fine-grained instructions and knowledge-intensive editing tasks. The visualizations illustrate how our method achieves superior instruction following while maintaining visual quality and source consistency.

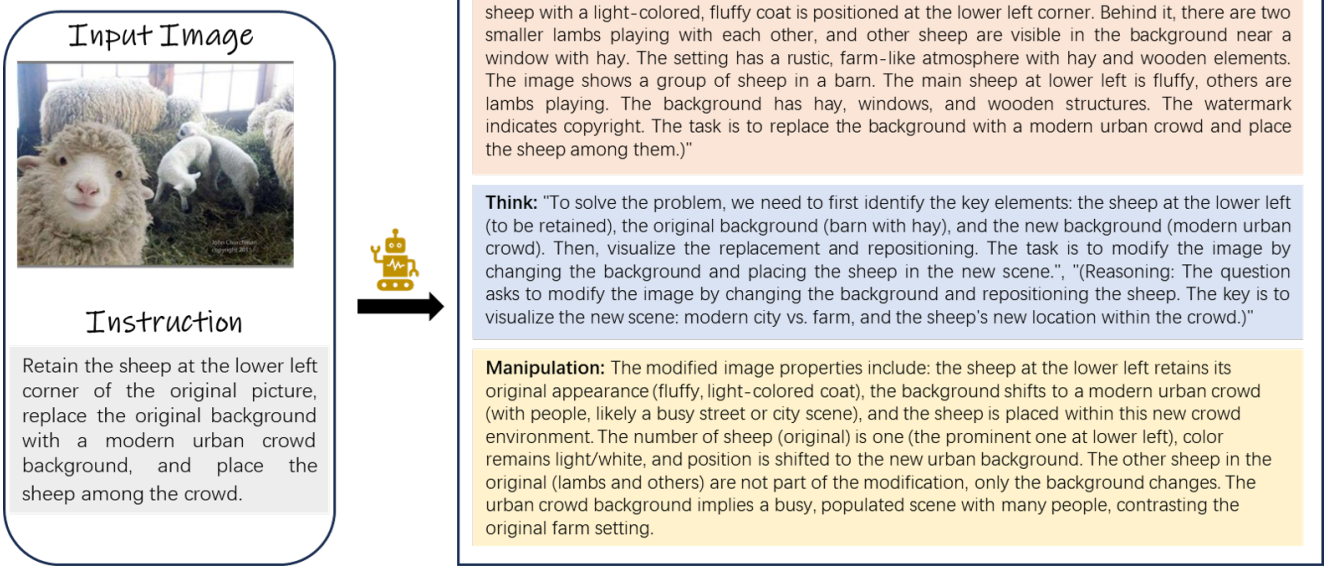
Compare to Instruction-based methods. Figure 9 presents qualitative comparisons between CogniEdit and state-of-the-art instruction-based editing methods: Qwen-Image-Edit-r1, Step1X-Edit, and OmniGen2. These examples highlight scenarios requiring precise attribute control, including exact color specifications, numerical quantities, and spatial positioning.

The visual comparisons reveal critical differences in fine-grained instruction following capabilities. CogniEdit consistently captures precise specifications that baseline methods fail to reproduce. Step1X-Edit and OmniGen2 exhibit particular difficulty with numerical attributes and spatial positioning, often ignoring these fine-grained details in favor of general semantic understanding. Qwen-Image-Edit-r1 performs better but still shows inconsistent handling of multi-attribute instructions. These results validate that our Dense GRPO Optimization and Dynamic Token Focus Relocation mechanism effectively enhance fine-grained attribute control.

Comparison with Knowledge-based Methods. Figure 10 compares CogniEdit against closed-source commercial VLM systems (Gemini 2, Doubao, GPT-4O) and chain-of-thought reasoning methods (Uni-CoT, UniWorld). These examples focus on knowledge-intensive editing tasks requiring domain expertise, factual knowledge, or complex



(a) Sample 1.



(b) Sample 2.

Figure 13. Samples of MLLM-enhanced instruction.

reasoning about visual semantics. We further provide the editing results of our method on Kris-Bench in Figure 11.

The knowledge-based comparisons highlight the importance of balancing semantic correctness with source fidelity. Commercial VLM systems like GPT-4O and Gemini 2 demonstrate strong instruction understanding and can generate semantically plausible content, but they often treat editing as a generation task, leading to poor source consistency—edited images may exhibit different lighting, style, or unnecessary modifications to unedited regions. Doubao shows similar issues with additional artifacts in complex scenes. Chain-of-thought methods (Uni-CoT, UniWorld) improve reasoning capabilities through explicit decomposition, but their editing quality remains inconsistent, particularly for instructions requiring both reasoning and precise visual control. In contrast, CogniEdit achieves superior balance: the MLLM-based reasoning component ensures semantic correctness and factual knowledge integration, while Dense GRPO Optimization maintains source consistency and visual quality.

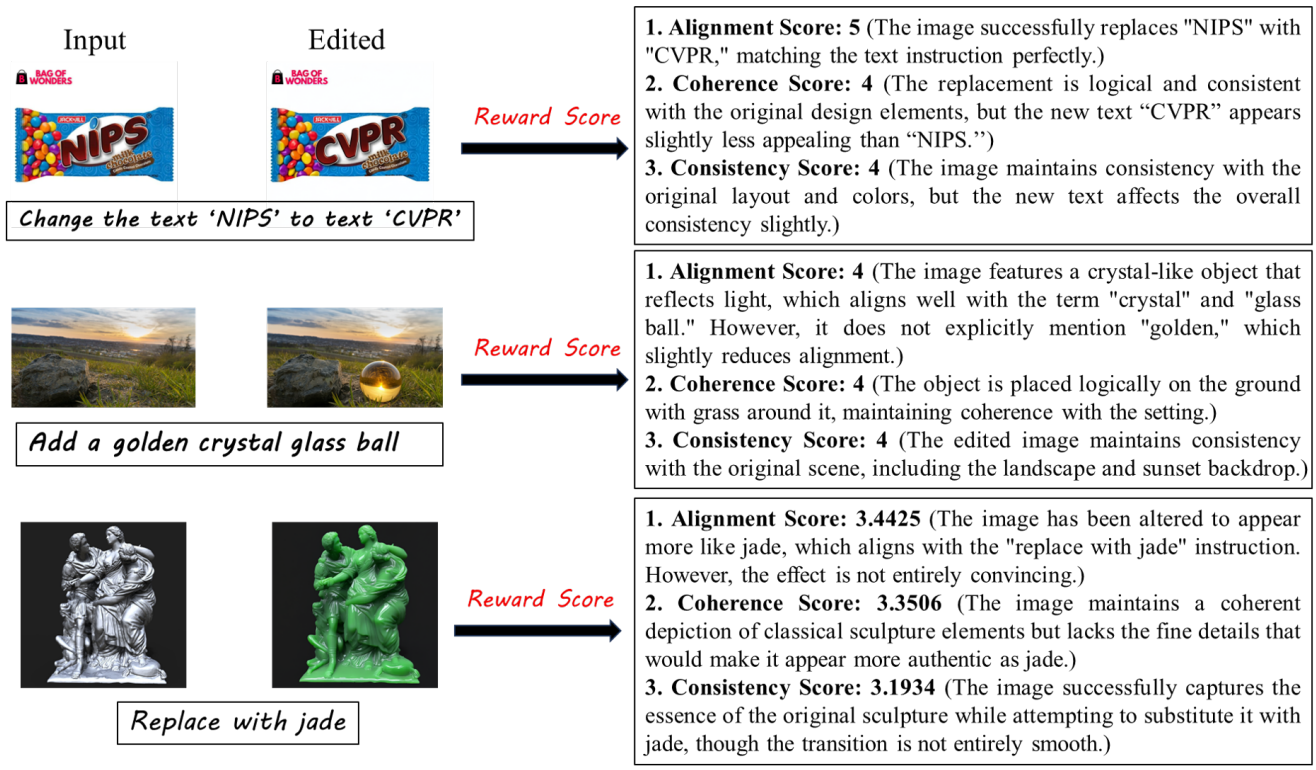


Figure 14. The samples of how the reward model evaluates the edited image.

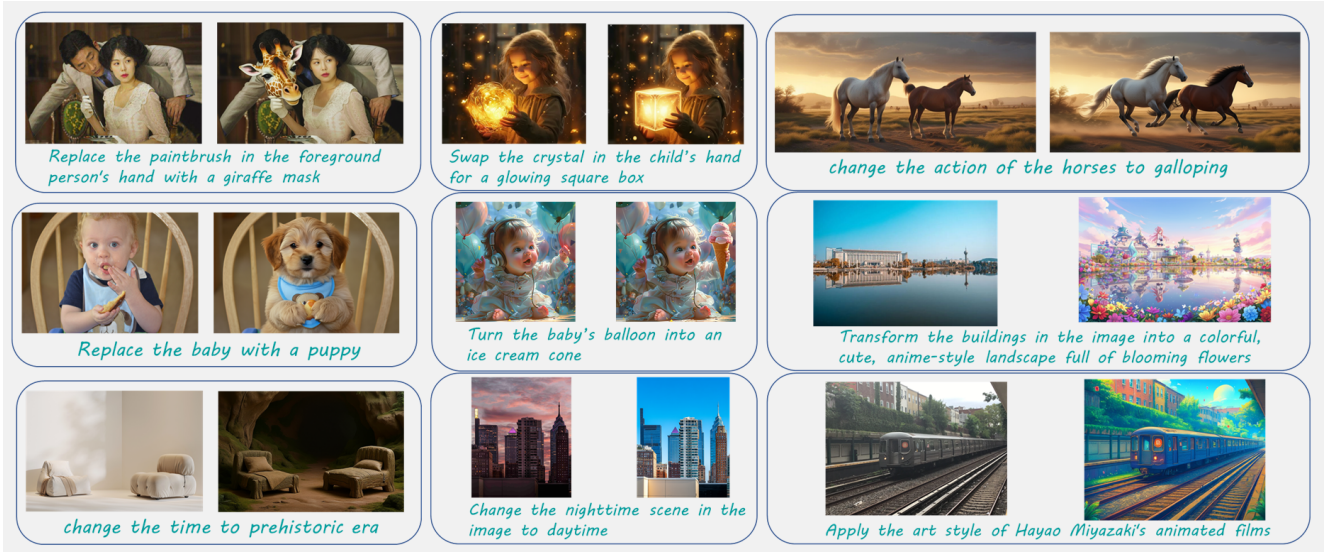


Figure 15. The editing results of our method on the GEdit-Bench.

11.4. User Study

To assess perceptual quality and editing fidelity from a human perspective, we conduct a comprehensive user study comparing CogniEdit against state-of-the-art instruction-based editing methods and commercial VLM systems. We recruit 10 participants with diverse backgrounds in computer vision and image editing. Each participant evaluates a series sampled editing pairs from Kris-Bench, ensuring broad coverage of knowledge-intensive scenarios.

Evaluation Protocol. Participants rate each edited image on four criteria using a 5-point Likert scale (1=poor, 5=ex-

cellent):

- **Editing Quality**: Visual plausibility, realism, and absence of artifacts in edited regions.
- **Instruction Adherence**: Alignment between the edited result and the textual instruction, including fine-grained attributes.
- **Consistency**: Preservation of unedited regions and coherence between edited and original content.
- **Overall Score**: Holistic assessment considering all aspects.

The evaluation is double-blind—participants are unaware of which method generated each result, and samples are presented in random order. For each editing task, we compute the mean score across all participants, and overall performance is obtained by averaging across all editing tasks.

For instruction-based editing methods. Table 4 compares CogniEdit against specialized instruction-based editing methods. Our approach achieves the highest scores across all criteria: editing quality (3.71), instruction adherence (3.49), consistency (3.60), and overall performance (3.47). Notably, CogniEdit outperforms the strong baseline Qwen-Image-Edit-r1 by significant margins—7.1% in editing quality and 28.4% in instruction adherence—demonstrating the effectiveness of dense GRPO optimization for fine-grained control.

Table 4. The results of the user study for instruction-based editing methods.

Method	Editing Quality			
	Quality	Adherence	Consistency	Overall
Step1X-Edit	2.743	3.103	2.713	2.790
OmniGen2	2.903	1.984	3.040	2.606
Qwen-Image-Edit-r1	3.468	2.718	3.281	2.968
Ours	**3.712**	**3.487**	**3.603**	**3.465**

For knowledge-based editing methods. Table 5 compares CogniEdit against commercial VLM systems on knowledge-intensive editing tasks.

Table 5. The results of the user study for knowledge-based editing methods.

Method	Editing Quality			
	Quality	Adherence	Consistency	Overall
Gemini 2	3.021	3.387	2.868	3.215
GPT-4o	3.184	3.450	3.059	**3.503**
Ours	**3.712**	**3.48**7	**3.603**	3.465

CogniEdit significantly outperforms both GPT-4O and Gemini 2 in editing quality (3.71 vs. 3.18 and 3.02) and consistency (3.60 vs. 3.06 and 2.87). Interestingly, GPT-4o achieves the highest instruction adherence (3.45), slightly above our method, but suffers from poor consistency, revealing a fundamental trade-off in VLM-based editing: these systems excel at understanding instructions but struggle to maintain source image fidelity during edits, as they treat editing more like generation tasks. Gemini 2's particularly weak consistency suggests even stronger generation bias. In contrast, CogniEdit achieves balanced performance across all metrics, validating that MLLM-based reasoning combined with dense reward optimization successfully preserves source consistency while following complex instructions. The overall scores (3.47 for CogniEdit vs. 3.50 for GPT-4o) are competitive, but our superior editing quality and consistency make CogniEdit more suitable for practical editing applications where source preservation is critical.