Supplementary Material - From Scale to Speed: Adaptive Test-Time Scaling for Image Editing

Xiangyan Qu[12*†], **Zhenlong Yuan**[3*], **Jing Tang**[3‡], **Rui Chen**[3], **Datao Tang**[3], **Meng Yu**[3], **Lei Sun**[3], **Yancheng Bai**[3], **Xiangxiang Chu**[3], **Gaopeng Gou**[12§], **Gang Xiong**[12], **Yujun Cai**[4],

[1]Institute of Information Engineering, Chinese Academy of Sciences [2]School of Cyber Security, University of Chinese Academy of Sciences
[3]AMAP, Alibaba Group [4]University of Queensland
[*] Equal contribution. [‡] Project lead. [§] Corresponding author.

In this appendix, we provide comprehensive supplementary material to offer a more complete understanding of our ADE-CoT framework. The appendix is organized as follows:

- Sec. A revisits and expands upon the motivation behind our work.
- Sec. B details the technical components of ADE-CoT.
- Sec. C presents additional experimental results and ablation studies.
- Sec. D outlines the limitations and directions for future research.
- Sec. E reviews extended related work in the field of test-time scaling and image editing.

Detailed contents are listed as follows:

[†]Work done during the internship at AMAP, Alibaba Group.

A. Further Discussion on Motivation

A.1. Limitations of SOTA Models on Complex Edits

Recent image editing models [8, 17, 24, 49] based on latent-level fusion between MLLMs and diffusion decoders have demonstrated impressive capabilities on standard editing tasks. However, their performance remains challenging when faced with complex editing scenarios:

Large pose changes. As shown in Fig. 1, baseline models (*i.e.*, default single-pass inference setting) struggle with edits requiring significant pose or action modifications. For instance, when asked to "`change the man's gesture to raising his hands`", the baseline unintentionally alters the surrounding context, including the position of the chair and the man's location in the scene. Similarly, the instruction "`make the action of the plane to taking off`" results in the baseline replacing the original plane with a completely different aircraft model and color scheme. The instruction "`change the bird's action to flapping its wings and flying`" often yields anatomically incorrect wing positions.

Multi-object modification. Complex edits involving multiple objects pose additional challenges for baseline models. In Fig. 1, we observe failures in instructions such as "`remove the woman standing next to the lady in white`", where the model fails to remove the correct person and leaves visible artifacts. Similarly, "`remove the foreground snow-covered trees`" results in incomplete removal, with only some trees being eliminated while others remain.

Fine-grained regional edits. Baseline models often fail to perform precise localized modifications. As shown in Fig. 1, instructions such as "`change clothes of the person in lower right corner to green`" frequently result in incorrect region selection and color bleeding to adjacent areas. The edit "`change the hair of the green-eyed toy figure to blonde`" shows similar issues with precise attribute modification. Fine-grained color changes such as "`alter the color of the front bus to lime`" or "`change the color of couch to yellow`" commonly affect unintended areas.

Multi-turn editing. As illustrated in Fig. 2, multi-turn edits are particularly susceptible to cascading errors, where mistakes in early turns propagate and accumulate through subsequent editing steps. For instance, in the first example, the initial instruction is "`Turn 1: Change the pants color to black.`" However, the model fails to execute this correctly, leaving the pants unchanged. This initial mistake cascades: subsequent edits are performed on an already flawed image. Consequently, the final result fails to reflect the cumulative user intent.

These limitations demonstrate that single-pass inference is insufficient for complex editing scenarios.

A.2. Image Chain-of-Thought Methods

Image Chain-of-Thought (Image-CoT) [26, 51, 57] offers a promising approach to address these challenges. As a test-time scaling strategy, Image-CoT generates multiple candidates through extended inference time. By selecting the best one from diverse candidates, it improves editing quality on complex scenarios without requiring additional training.

Best-of-N (BoN) [26, 51] is the standard method for Image-CoT. As shown in Fig. 3(a) and summarized in Alg. 1, it consists of two primary stages: ❶ **Generation**. This stage produces a diverse set of N candidate images. This is achieved through a loop that iterates N times (Line 2). Within each iteration, diversity is introduced by sampling a unique initial noise $x_T^{(i)}$ (Line 3) and optionally rewriting the text prompt c into a variant $c^{(i)}$ (Line 4). The core of this stage is the `Sampler` function (Line 5), which performs a complete denoising process from timestep T down to 0 to generate a clean latent representation $x_0^{(i)}$. This latent is then decoded into the final image $I^{(i)}$ (Line 6). ❷ **Selection**. For each image $I^{(i)}$, a general MLLM verifier $\texttt{Vrf}_g$ computes a score $S^{(i)}$ that reflects its quality and adherence (Line 7). The $(I^{(i)}, S^{(i)})$ pair is added to the set $\mathcal{U}$ (Line 8). After all N candidates are generated, the candidate with the highest score is chosen as the final output I^* (Line 10). **Computational cost**. While BoN improves generation quality, its computational cost scales linearly with the number of samples N. Since every candidate must complete the full T denoising steps before selection, the total cost is $N \times T$ function evaluations (NFE). This inefficiency makes BoN impractical for large-scale sampling.

Early pruning strategies in Image-CoT. To address the inefficiency of BoN, early pruning [23, 57, 59] is a standard approach. The core idea is to identify and discard low-potential candidates at an early stage, avoiding the cost of their full generation. A unified framework for two common strategies is shown in Alg. 2, controlled by the `mode`. ❶ **Early preview via additional denoising steps**. This variant, proposed by TTS-EF [59], generates preview images by performing additional denoising from t_e to 0 (Line 7). The `Sampler` function produces a complete denoised latent $x_{t_e}^{(i)}$ that is decoded into a clear preview image $I_{t_e}^{(i)}$ (Line 12). This provides high-quality previews for reliable verification. However, it introduces extra computational cost of t_e denoising steps per candidate. After pruning, passing candidates resume denoising denoising from T to 0 to generate the final output (Line 17). ❷ **Early pruning on intermediate states**. This variant, used by PRM [57] and PARM [57], samples from T to t_e to obtain an intermediate latent $x_{t_e}^{(i)}$ (Line 10). The latent is directly decoded into a preview image. This approach requires no extra steps for preview generation. After pruning, passing candidates resume denoising from t_e to 0 to complete generation (Line 19). However, the preview may be noisy due to incomplete denoising, which may affect verification accuracy.

A.3. Issues of Image-CoT Methods for Editing

Most Image-CoT methods [21, 26, 57, 59] are developed for text-to-image generation. However, directly applying them to editing is suboptimal. As discussed in the Introduction, this mismatch causes three issues:

Inefficient resource allocation. Image-CoT methods typically use a fixed sampling budget for all edits, which may be inefficient. To validate this, we measure the performance gain relative to edit difficulty. First, we defined edit difficulty by generating a single image for each editing instance and using its MLLM score as an initial score. We then grouped the edit tasks into bins based on this initial score. For all tasks, we applied a standard Best-of-32 (BoN) sampling strategy. We calculated the average score gain between the initial score and the final score after adding Image-CoT for each bin. As shown in Fig. 3, edits with high initial scores (simple edits) showed minimal improvement from large-scale sampling. In contrast, edits with low initial scores (complex edits) benefited significantly. This confirms that a fixed budget wastes computational resources on simple edits that do not require extensive sampling.

Unreliable early-stage verification. Current Image-CoT methods [23, 57, 59] use general MLLM scores to prune candidates at early denoising stages. We examine whether these scores correctly identify high-potential candidates. In our experiment, we generate $N = 32$ candidates

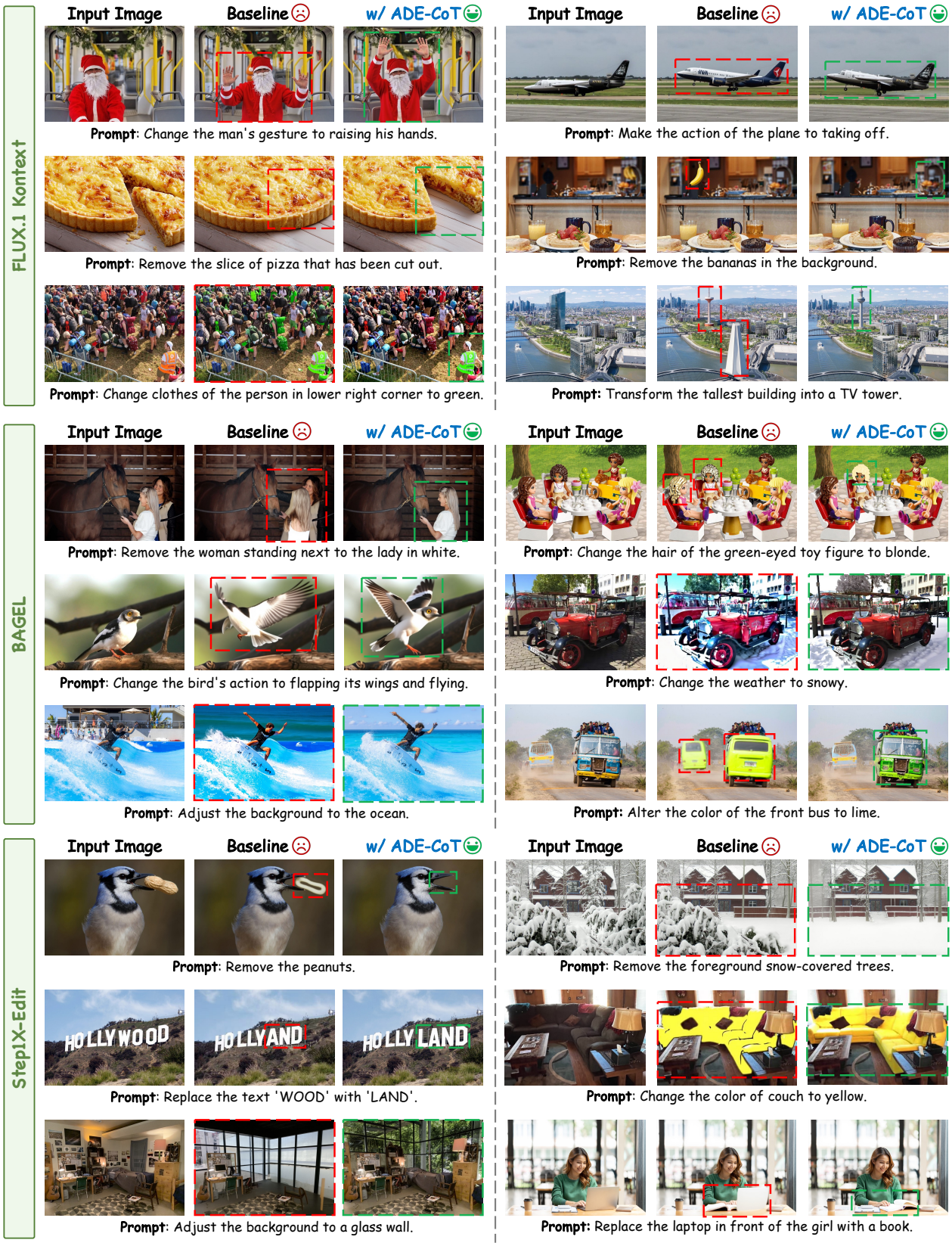


Figure 1. **Qualitative comparison on complex edits.** We compare three SOTA editing models (FLUX.1 Kontext [17], BAGEL [8], and Step1X-Edit [24]) on challenging edits (large pose changes, multi-object modifications, and fine-grained regional edits). Baseline models often fail, while our ADE-CoT produces correct results via adaptive test-time scaling. 🔍 **Zoom in** for detailed view.

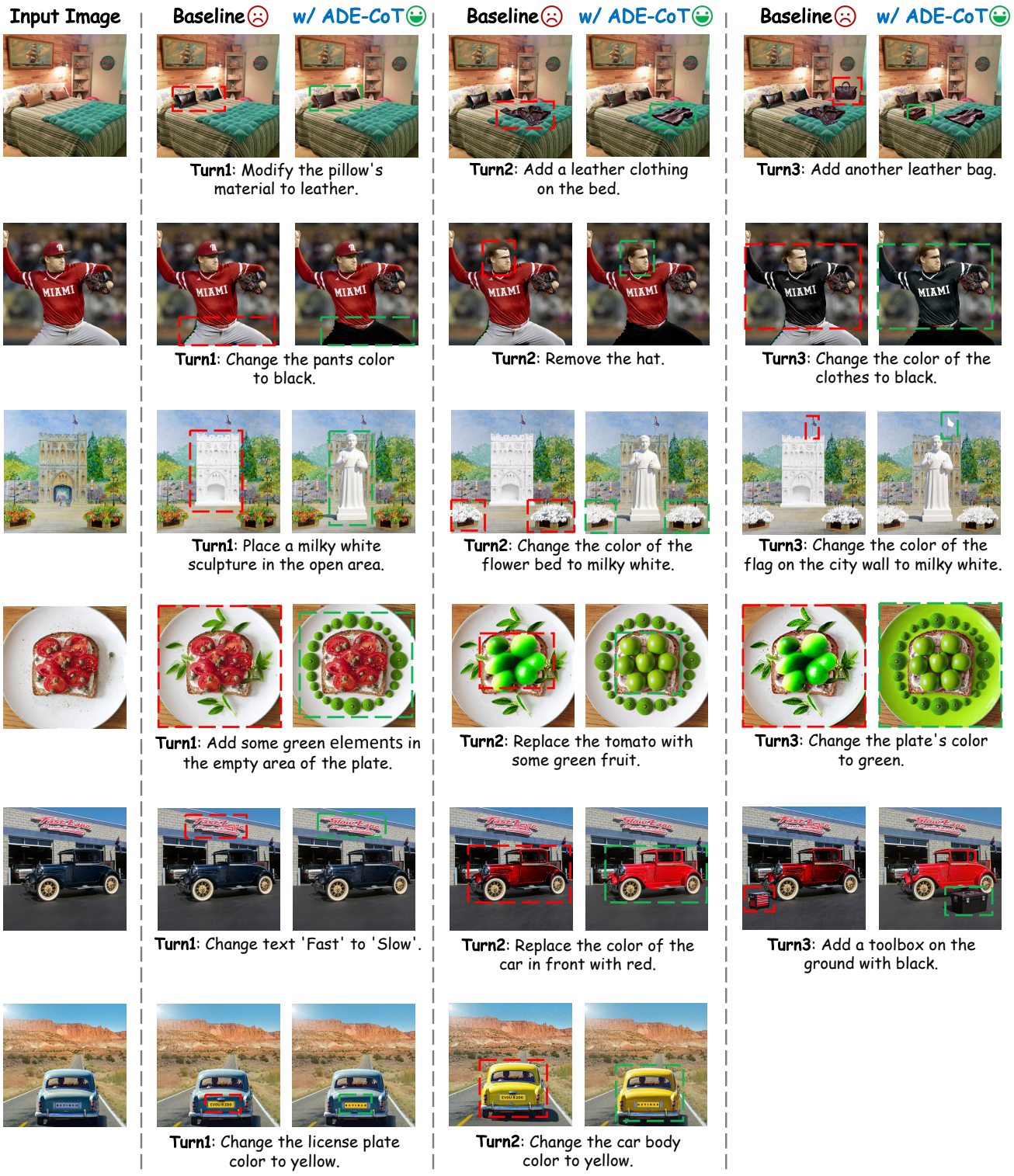


Figure 2. **Qualitative comparison on multi-turn edits**. The baseline model often fails to preserve the context from previous edits, leading to accumulated errors in subsequent turns. Our ADE-CoT maintains consistency across multiple sequential instructions, producing correct final images that reflect all requested changes. 🔍 **Zoom in** for detailed view.

Algorithm 1 *Best-of-N* (BoN) algorithm for image editing.

Require: source image I_{src}, text prompt c, number of samples N, and total steps T

$\mathcal{U} \leftarrow \{\}$ ▷ Initialize empty set for (I, S) pairs
for $i = 1$ to N **do**
  $x_T^{(i)} \sim \mathcal{N}(0, \mathrm{I})$ ▷ Sample initial noise
  $c^{(i)} \leftarrow \texttt{Rewrite}(c)$ ▷ (optional) Rewrite prompt
  $x_0^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T^{(i)}, c^{(i)}, T, 0)$ ▷ Sample from T to 0, *i.e.*, full denoising process
  $I^{(i)} \leftarrow \texttt{VAE\_Decoder}(x_0^{(i)})$
  $S^{(i)} \leftarrow \texttt{Vrf}_g(I_{\text{src}}, I^{(i)}, c)$ ▷ Compute general MLLM score
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{(I^{(i)}, S^{(i)})\}$ ▷ Update set
end for
$I^* \leftarrow \arg\max_{(I,S)\in\mathcal{U}} S$
return I^*

Algorithm 2 *Early Pruning* algorithm for image editing.

Require: source image I_{src}, text prompt c, number of samples N and steps T, early step t_e, reject threshold S_{rj}, $\texttt{mode} \in$ {'additional_steps', 'intermediate_state'}

$\mathcal{U} \leftarrow \{\}$ ▷ Initialize empty set for (I, S) pairs
for $i = 1$ to N **do**
  $x_T^{(i)} \sim \mathcal{N}(0, \mathrm{I})$ ▷ Sample initial noise
  $c^{(i)} \leftarrow \texttt{Rewrite}(c)$ ▷ (optional) Rewrite prompt
  if mode == 'additional_steps' **then**
    ▷ TTS-EF [59] method
    $x_{t_e}^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T^{(i)}, c^{(i)}, t_e, 0)$ ▷ Sample from t_e to 0, *i.e.*, early preview by additional denoising steps
  else if mode == 'intermediate_state' **then**
    ▷ PRM [57] and PARM [57] methods
    $x_{t_e}^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T^{(i)}, c^{(i)}, T, t_e)$ ▷ Sample from T to t_e, *i.e.*, partially denoising process
  end if
  $I_{t_e}^{(i)} \leftarrow \texttt{VAE\_Decoder}(x_{t_e}^{(i)})$
  $S_{t_e}^{(i)} \leftarrow \texttt{Vrf}_g(I_{\text{src}}, I_{t_e}^{(i)}, c)$ ▷ Compute general MLLM score
  ▷ Prune sample below S_{rj}
  if $S_{t_e}^{(i)} >= S_{\text{rj}}$ **then**
    if mode == 'additional_steps' **then**
      $x_0^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T^{(i)}, c^{(i)}, T, 0)$ ▷ Sample from T to 0, *i.e.*, full denoising process
    else if mode == 'intermediate_state' **then**
      $x_0^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_{t_e}^{(i)}, c^{(i)}, t_e, 0)$ ▷ Sample from t_e to 0, *i.e.*, resume denoising process
    end if
    $I^{(i)} \leftarrow \texttt{VAE\_Decoder}(x_0^{(i)})$
    $S^{(i)} \leftarrow \texttt{Vrf}_g(I_{\text{src}}, I^{(i)}, c)$ ▷ Compute general MLLM score
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{(I^{(i)}, S^{(i)})\}$ ▷ Update set
  end if
end for
$I^* \leftarrow \arg\max_{(I,S)\in\mathcal{U}} S$
return I^*

per edit case and evaluate each at an early timestep $t_e = 8$ using VIE-Score [16]. Candidates scoring below a rejection threshold S_{rj} are *pruned*. We then complete the full denoising process for all candidates to obtain final scores. As shown in Fig. 4, this misjudgement occurs consistently across all tested models and datasets. On average, 40% of the pruned samples ultimately achieve high final scores (≥ 6). This indicates that general scores incorrectly discard many high-potential candidates during early pruning. This misjudgement leads to degraded final performance.

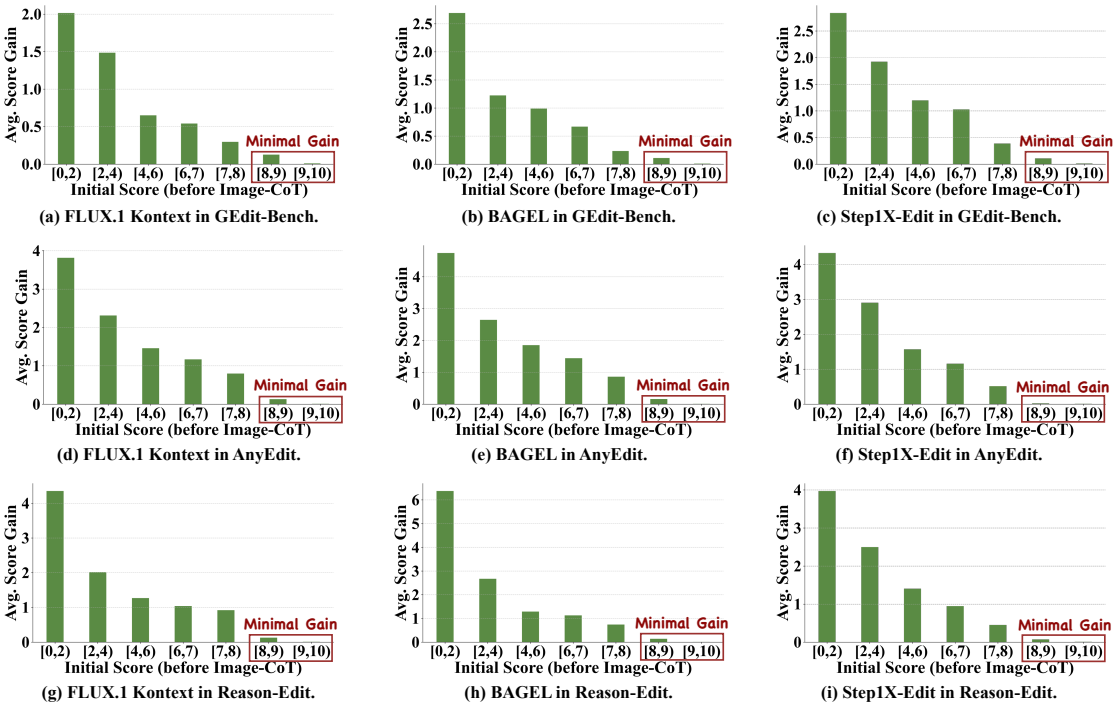


Figure 3. **Inefficient resource allocation with fixed sampling budgets**. This figure extends the analysis from Fig. 2(a) to three SOTA models (FLUX.1 Kontext, BAGEL, and Step1X-Edit) and three benchmarks (GEdit-Bench, AnyEdit, and Reason-Edit). Edits with high initial scores (**red boxes**, [8,9) and [9,10)) show minimal improvement across all three models and three benchmarks. Edits with low initial scores (indicating complex tasks) benefit significantly from Image-CoT, achieving substantial performance gains. This demonstrates that fixed sampling budgets waste computation on simple edits, motivating our difficulty-aware resource allocation strategy.

Redundant edited results. The goal-directed nature of image editing suggests that large-scale sampling may produce many similar, correct results. To quantify this redundancy, we apply a Best-of-32 (BoN) strategy to each editing instance. For each case, we identify the best score achieved among the 32 candidates and then count how many candidates share the same best score. As shown in Fig. 5, we observe this redundancy across three models and three datasets. For edit cases with high final scores (*e.g.*, in the range $[7, 9)$), a large number of candidates, often more than 8, achieve the identical best score. Since only one intent-aligned result is sufficient for editing, this redundancy reflects unnecessary computation. Existing breadth-first search strategies [26, 57, 59] generate all candidates before selection. This leads to wasted denoising steps on redundant correct outputs.

To address these issues, we propose ADE-CoT, an adaptive test-time scaling framework for image editing. Our method improves editing performance while maintaining computational efficiency. Specifically, we introduce three key strategies: (1) difficulty-aware resource allocation to dynamically adjust sampling budgets based on edit difficulty, (2) edit-specific verification to accurately identify high-potential candidates during early pruning, and (3) depth-first opportunistic stopping to reduce redundant computation on correct results. As shown in Fig. 1 and Fig. 2, our method significantly enhances baseline performance on complex editing scenarios discussed in Sec. A.1.

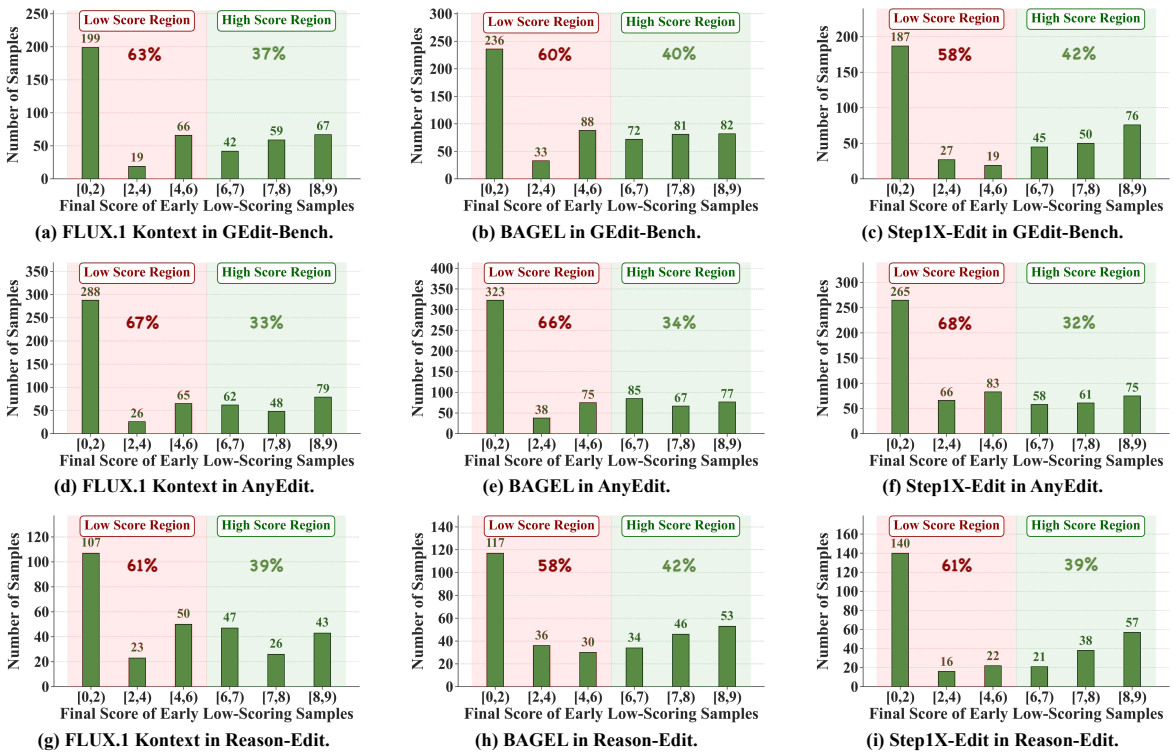


Figure 4. **Misjudgement by general MLLM scores in early pruning.** This figure extends the analysis from Fig. 2(b) to three SOTA editing models (FLUX.1 Kontext, BAGEL, and Step1X-Edit) evaluated on three benchmarks (GEdit-Bench, AnyEdit-Test, and Reason-Edit). Samples with low early scores are categorized by their final scores (x-axis): low score region (**red**, [0,6)) and high score region (**green**, [6,9)). On average, 37% of early low-scoring samples eventually achieve high final scores, yet would be incorrectly discarded by general MLLM scores. This demonstrates unreliable early-stage verification in editing, motivating our edit-specific verification strategy.

B. Details of the Proposed Method ADE-CoT

In this section, we provide implementation details of three components in ADE-CoT: difficulty-aware resource allocation (Sec. B.1), edit-specific verification in early pruning (Sec. B.2), and depth-first opportunistic stopping (Sec. B.3).

B.1. Difficulty-aware Resource Allocation

As described in Sec. 3.1 of the main paper, our difficulty-aware resource allocation strategy dynamically adjusts the sampling budget to improve computational efficiency. The process is summarized in Alg. 4. It first generates a single candidate to estimate the edit difficulty, which then determines the final sampling budget, N_a.

❶ The process begins by generating one preliminary-estimation image (Lines 1-3). First, we sample an initial noise vector x_T from a standard normal distribution (Line 1). A diffusion `Sampler` then generates a clean latent x_0 from this noise, conditioned on the source image I_{src} and instruction c (Line 2). A VAE decoder subsequently converts the latent x_0 into a pixel-space image I (Line 3).

❷ Next, we estimate the edit difficulty using this initial image (Line 4). A general MLLM verifier, $\texttt{Vrf}_g$, evaluates the image I to produce an initial score S. This score acts as a proxy for difficulty, where a high score suggests an easy edit and a low score indicates a difficult one.

❸ The adaptive budget N_a is then calculated based on this score (Line 5), following Eq. 3 from the main text. For an easy edit where the score S is high, the budget N_a is reduced towards the minimum budget N_{min}. Conversely, for a difficult edit where S is low, the budget N_a increases towards the original budget N. The hyperparameter γ controls the sensitivity of this adjustment, and the ceiling function $\lceil \cdot \rceil$ ensures the budget is an integer. Finally, the algorithm returns the calculated budget N_a (Line 6). This strategy effectively allocates more computational resources to difficult cases and saves them on easy ones.

B.2. Edit-specific Verification in Early Pruning

As described in Sec. 3.2 of the main paper, our edit-specific verification strategy addresses the misjudgement issue of general MLLM scores in early denoising stages. The detailed process is summarized in Alg. 5.

The algorithm operates as follows. ❶ We first initialize empty sets for intermediate latents $\mathcal{X}_{t_e}$, prompts $\mathcal{C}$, and scores $\mathcal{S}_{t_e}$ (Line 1). ❷ For each of the N_a samples (Lines 2-13), we sample random noise $x_T^{(i)} \sim \mathcal{N}(0, \mathrm{I})$ (Line 3)

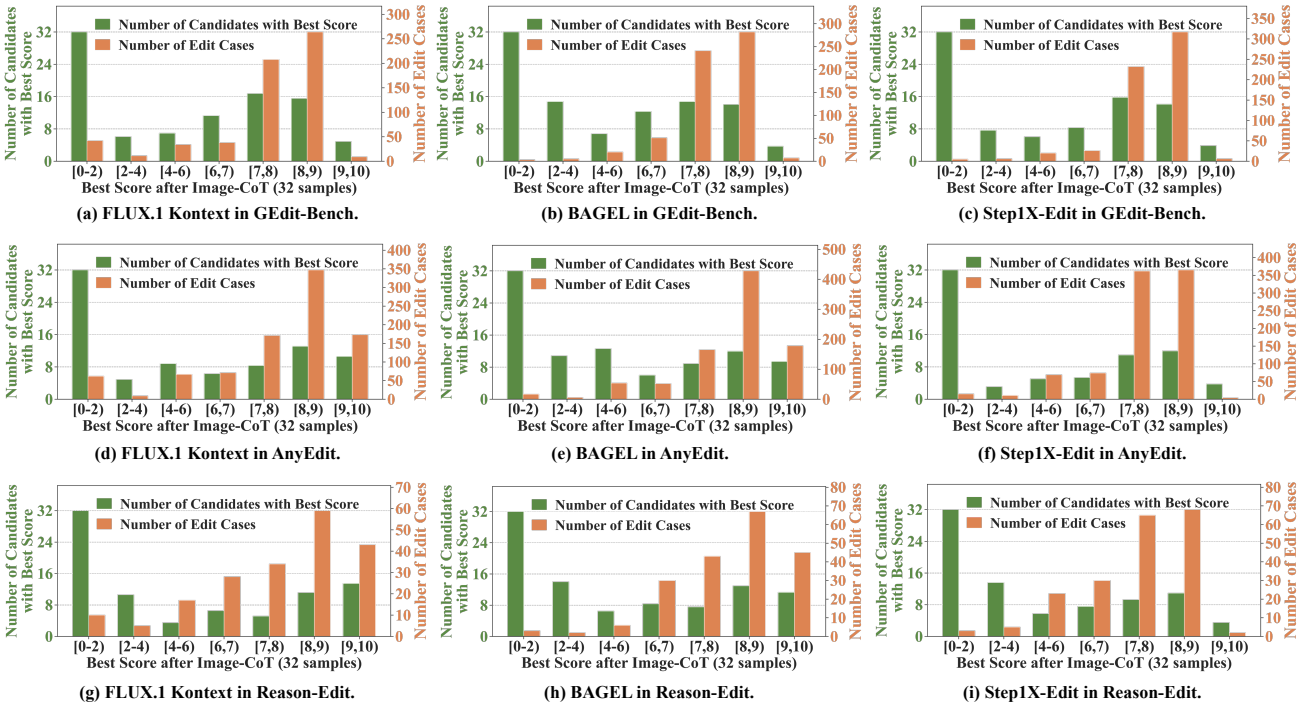


Figure 5. **Redundant edited results in large-scale sampling.** This figure extends the analysis from Fig. 2(c) to three SOTA editing models (FLUX.1 Kontext, BAGEL, and Step1X-Edit) evaluated on three benchmarks (GEdit-Bench, AnyEdit-Test, and Reason-Edit). For most edit cases, a large number of candidates (green bars, left y-axis) share identical best scores. The number of edit cases exhibiting such redundancy (orange bars, right y-axis) increases significantly in high score regions, particularly in [7,8) and [8,9) (x-axis). This demonstrates that Image-CoT produces redundant correct outputs in goal-directed editing, motivating our opportunistic stopping strategy.

Algorithm 3 Our ***ADaptive Edit-CoT*** (`ADE-CoT`) algorithm for image editing.

Require: source image I_{src}, text prompt c, number of samples N and steps T, early step t_e and retain step t_l

1: ▷ **Adapt N by edit difficulty**
2: $N_a \leftarrow \texttt{Adapt\_Num}(I_{\text{src}}, c, N, T)$
3: ▷ **Sample from T to t_e and prune samples**
4: $\mathcal{X}_{t_e}, \mathcal{C} \leftarrow \texttt{Early\_Prune}(I_{\text{src}}, c, N_a, T, t_e)$
5: ▷ **Sample from t_e to t_l and retain top results; Sample from t_l to 0 and select intent-aligned results**
6: $\mathcal{U} \leftarrow \texttt{Adaptive\_Stop}(I_{\text{src}}, c, \mathcal{X}_{t_e}, \mathcal{C}, t_e, t_l)$
7: $I^* \leftarrow \arg\max_{(I,S)\in\mathcal{U}} S$
8: **return** I^*

and optionally rewrite the prompt $c^{(i)}$ (Line 4). We then perform denoising from timestep T to early timestep t_e (Line 5) and apply the one-step preview mechanism to obtain $x^{(i)}_{0|t_e}$ (Line 6), which is decoded into preview image $I^{(i)}_{0|t_e}$ (Line 7). The unified score $S^{(i)}_{0|t_e}$ is computed by combining general MLLM score, edited-region correctness, and instruction-caption consistency (Line 8). Candidates with scores below the rejection threshold S_{rj} are pruned (Lines 10-12), while others are retained. ❸ After processing all samples, we remove visually similar candidates using DINOv2 features and the threshold τ_{sim} (Lines 14-15). ❹ Finally, the remaining candidates are sorted by their unified scores in descending order (Lines 16-17), which guides the subsequent depth-first generation stage.

B.2.1. One-Step Preview Mechanism

Lines 6-7 of Alg. 5 implement the one-step preview mechanism, which obtains approximate previews without additional denoising steps. To validate that these early previews reliably reflect the final output, we visualize the process in Fig. 6. The figure compares the noisy latent I_t with our corresponding one-step preview $I_{0|t}$ across various timesteps for three models. As shown, our preview generates a clear and high-fidelity approximation of the final result, even at very early stages where the noisy latent is uninterpretable (*e.g.*, $t = 8$). This observation confirms that our one-step preview provides a sufficiently clear signal for early-stage

Algorithm 4 `AdaptNum` algorithm - Lines 1-2 of Alg. 3.

Require: source image I_{src}, text prompt c, number of samples N and steps T
Require: number of minimum samples $N_{\min}$, maximum possible score $S_{\max}$, sensitivity factor γ
$x_T \sim \mathcal{N}(0, \mathrm{I})$
$x_0 \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T, c, T, 0)$ ▷ Sample from T to 0
$I \leftarrow \texttt{VAE\_Decoder}(x_0)$
$S \leftarrow \texttt{Vrf}_g(I_{\text{src}}, I, c)$ ▷ Compute general MLLM score
$N_a \leftarrow N_{\min} + \lceil (N - N_{\min}) \times (1 - S/S_{\max})^{\gamma} \rceil$ ▷ Adapt N based on S (*cf*. Sec 3.1 Eq. 3)
return N_a

Algorithm 5 `Early_Prune` algorithm - Lines 3-4 of Alg. 3.

Require: source image I_{src}, text prompt c, number of samples N_a and steps T, early step t_e
Require: reject threshold S_{rj}, similarity threshold τ_{sim}
$\mathcal{X}_{t_e} \leftarrow \{\}, \mathcal{C} \leftarrow \{\}, \mathcal{S}_{t_e} \leftarrow \{\}$ ▷ Initialize empty set
for $i = 1$ to N_a **do**
  $x_T^{(i)} \sim \mathcal{N}(0, \mathrm{I})$
  $c^{(i)} \leftarrow \texttt{Rewrite}(c)$ ▷ (optional) Rewrite prompt
  $x_{t_e}^{(i)} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_T^{(i)}, c^{(i)}, T, t_e)$ ▷ Sample from T to t_e
  $x_{0|t_e}^{(i)} \leftarrow \texttt{One\_Step\_Preview}(x_{t_e}^{(i)}, t_e)$ ▷ Preview image from intermediate latent (*cf*. Sec 3.2 Eq. 4)
  $I_{0|t_e}^{(i)} \leftarrow \texttt{VAE\_Decoder}(x_{0|t_e}^{(i)})$
  $S_{0|t_e}^{(i)} \leftarrow \texttt{Vrf}(I_{\text{src}}, I_{0|t_e}^{(i)}, c)$ ▷ Compute unified score (*cf*. Sec 3.2, Eq. 8)
  ▷ **Filter error by evaluated score**
  if $S_{0|t_e}^{(i)} >= S_{\text{rj}}$ **then**
    $\mathcal{X}_{t_e} \leftarrow \mathcal{X}_{t_e} \cup \{x_{t_e}^{(i)}\}, \mathcal{C} \leftarrow \mathcal{C} \cup \{c^{(i)}\}, \mathcal{S}_{t_e} \leftarrow \mathcal{S}_{t_e} \cup \{S_{0|t_e}^{(i)}\}$ ▷ Update set
  end if
end for
▷ **Filter visually similar candidates**
$\mathcal{X}_{t_e}, \mathcal{C}, \mathcal{S}_{t_e} \leftarrow \texttt{Remove\_Similar}(\mathcal{X}_{t_e}, \mathcal{C}, \mathcal{S}_{t_e}, \tau_{\text{sim}})$
▷ **Sort by evaluated score**
$\mathcal{X}_{t_e}, \mathcal{C} \leftarrow \texttt{Sort\_by\_Score}(\mathcal{X}_{t_e}, \mathcal{C}, \text{key} = \mathcal{S}_{t_e})$ ▷ Sort $\mathcal{X}_{t_e}, \mathcal{C}$ by $\mathcal{S}_{t_e}$ in descending order
return $\mathcal{X}_{t_e}, \mathcal{C}$

evaluation, enabling accurate pruning.

B.2.2. General Score by MLLM

Following prior work [59], we use VIE-Score [16] as our general MLLM verifier. VIE-Score evaluates an edited image based on two criteria: Semantic Consistency (SC), which measures instruction adherence and preservation of unedited regions, and Perceptual Quality (PQ), which assesses visual realism and aesthetics. The final overall score is calculated as the geometric mean of these two components: $S_{\text{gen}} = \sqrt{S_{\text{SC}} \times S_{\text{PQ}}}$. While this general score provides a coarse-grained assessment, it struggles to detect subtle errors such as mislocalized edits or semantic misalignment in early denoising stages. To address this, we complement it with edit-specific metrics.

B.2.3. Edited-Region Correctness

In Fig. 7, we present the prompt P_{reg} used to identify the edited or kept objects. When the MLLM successfully identifies the edited object, the mask M corresponds to that object's region. When the MLLM identifies the kept object, the mask M is set to the inverted region (*i.e.*, the complement of the kept object). If the MLLM cannot accurately determine either the edited or kept objects, we skip this verifier. Since directly computing RGB differences across the entire image is computationally expensive, we employ a sliding window approach to aggregate the change map Δ. However, the obtained mask may not perfectly align with the true editing region. To address this, we apply an adaptive mask refinement strategy. If all early preview candidates yield $S_{\text{reg}} = 0$, we iteratively expand the mask M by padding additional pixels around its boundary. The expansion process continues until at least one candidate achieves

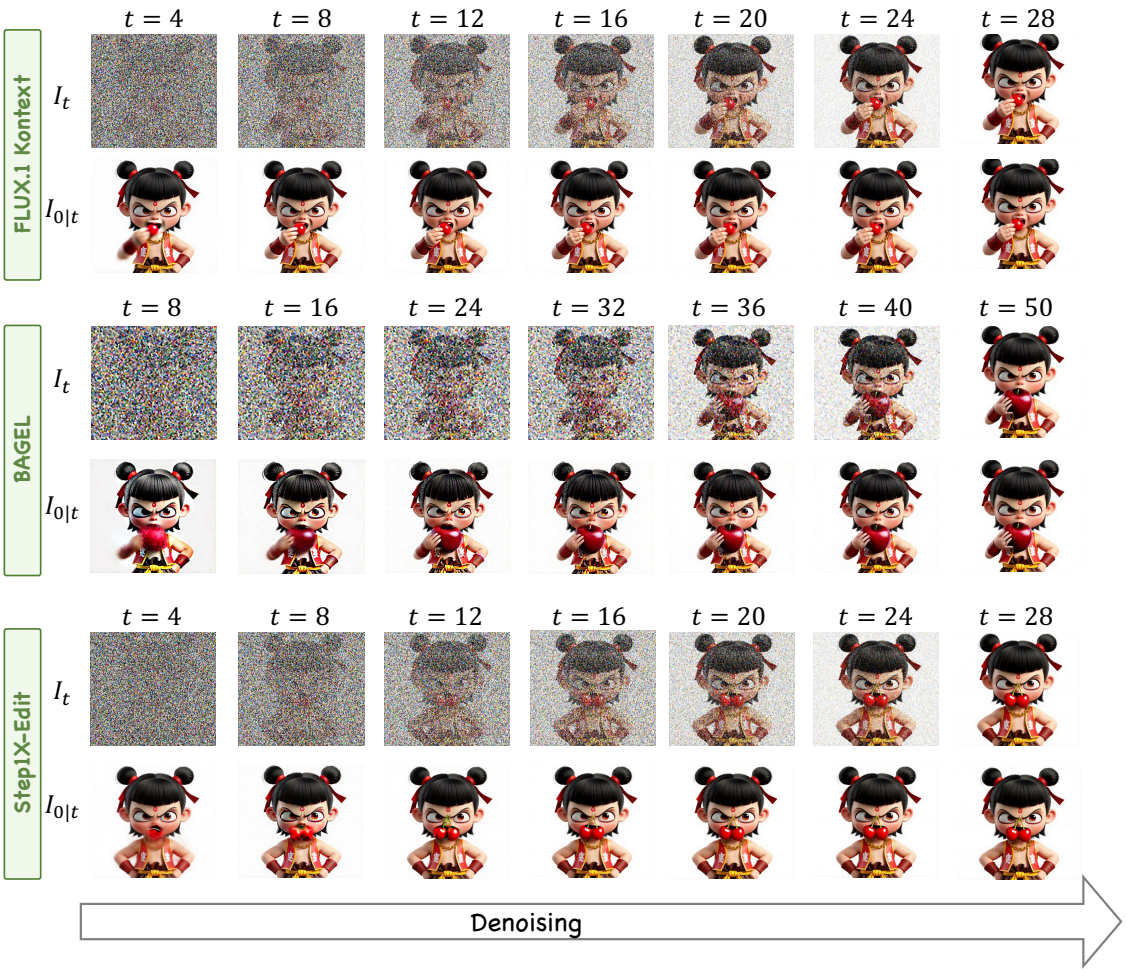


Figure 6. **Effectiveness of the one-step preview mechanism**. We compare the noisy latent I_t (top row for each model) with our corresponding one-step preview image $I_{0|t}$ (bottom row) at various denoising timesteps. Across all three models, our one-step preview generates a clear and high-fidelity approximation of the final output, even at very early stages (*e.g.*, $t = 8$). This demonstrates that the preview accurately reflects the final image's content and quality, providing a solid basis for our edit-specific verifiers.

$S_{\text{reg}} > 0$. This ensures that the mask adequately covers the actual edited region. This refinement step improves the robustness of region correctness evaluation.

B.2.4. Instruction-Caption Consistency

In Fig. 8, we present the prompt P_{cap}, which instructs an MLLM to generate a target caption for the ideally edited image. However, for subtle edits like local object modifications, this caption-based score may not be sufficiently discriminative. To ensure the reliability of the generated caption, we introduce a two-stage filtering process. First, we confirm that the MLLM correctly understands the source image. We check if its generated `original_caption` has a CLIP score above a threshold (default 0.27) with the source image. Second, we verify that the `edited_caption` actually reflects a change. We ensure its textual similarity to the `original_caption` is below a threshold (default 0.9). Only if a caption passes both these checks is it deemed reliable.

B.2.5. Filtering Visually Similar Candidates

To eliminate redundancy, we filter out visually similar candidates. Goal-directed image editing often yields multiple redundant results during large-scale sampling. Notably, this redundancy is already apparent in the early preview images, as illustrated in Fig. 9. To address this, we extract visual embeddings from each preview using DINOv2 and compute pairwise similarity. If the similarity between two candidates exceeds a threshold τ_{sim}, the one with the lower unified score is discarded. This step ensures that only visually distinct and high-potential candidates are retained.

B.3. Depth-first Opportunistic Stopping

As described in Sec. 3.3 of the main paper, our depth-first opportunistic stopping strategy avoids unnecessary computation on redundant yet correct results. It is summarized in

Prompt P_{reg} for identifying edited and keep objects

System Role: You are an assistant in an image-editing pipeline. Your sole task is to determine, from a textual edit instruction and the original image, (1) which object(s) in the image must be edited and (2) which object(s) are explicitly required to remain unchanged ("keep").

INPUT

The user will always provide both keys:

- `original image` – the image to be edited.
- `edit instruction` – the user's textual instruction that specifies how the image should be edited.

OUTPUT (must follow exactly)

Return one of the two following JSON structures:

Case A. The `edit_instruction` clearly identifies at least one concrete visual object to be edited:

```
{"edit_object": ["<the object(s) in the image that must be edited>"],
 "keep_object": ["<the object(s) that must NOT be edited>"]}
```

- Both keys must be present.
- If no "keep" object is mentioned, output an empty array for `keep_object`.
- If multiple distinct objects are explicitly requested, list each object as a separate string in the array.
- Regardless of whether the edit instruction is in English or Chinese, ALWAYS return object names in English.

Case B. No concrete visual object is to be edited (ambiguous, global change, object absent, or the instruction is about modifying, adding, or deleting TEXT in the image):

```
{"edit_object": null,
 "keep_object": null}
```

DECISION RULES

- "Clearly identified" means the instruction contains explicit nouns or noun phrases that unambiguously map to elements in the image description (e.g., "cat," "red cup," "man's face," "background sky").
- If the instruction applies to the entire image without pointing to a specific object (e.g., "add a vintage filter," "increase brightness"), return Case B.
- If the instruction references an object absent from the image description, return Case B.
- If the instruction's main purpose is to change, remove, or add TEXT in the image, return Case B (both values null).
- Do NOT perform the edit and do NOT add explanations—output the JSON only.

User Input:

`edit instruction:` ⟨edit_prompt⟩
`original image:` ⟨image⟩

Figure 7. **Prompt P_{reg} for identifying edited and keep objects**.

Alg. 6, which consists of two key components: (1) a late-stage filter to retain the most promising candidates; and (2) an instance-specific verifier to guide the stopping decision.

The algorithm operates as follows. ❶ We first initialize an empty set $\mathcal{U}$ to store image-score pairs (I, S), a retain threshold S_{rt}, and a high-score count N_{cnt} (Lines 1-3). ❷ For each candidate (x_{t_e}, c_r) from the sorted sets $(\mathcal{X}_{t_e}, \mathcal{C})$ (Line 5), we perform depth-first sequential generation. We first sample from early timestep t_e to late timestep t_l (Line 6) and apply the one-step preview mechanism to obtain $x_{0|t_l}$ (Line 7), which is decoded into preview image $I_{0|t_l}$ (Line 8). The unified score $S_{0|t_l}$ is computed using the same verifiers as in early pruning (Line 9). ❸ We apply an adaptive late-stage filter. If $S_{0|t_l} \geq S_{\text{rt}} - \delta$, we update the retain threshold S_{rt} (Line 12) and continue sampling from t_l to 0 to generate the final image I (Lines 13-14). Otherwise, we skip this candidate and proceed to the next one. ❹ For each retained candidate, we compute the unified score S (Line 15) and apply the instance-specific verifier to obtain S_{spec} (Line 17). We update the evaluated score as $S \leftarrow S + S_{\text{spec}}$ (Line 18). If $S_{\text{spec}} \geq S_{\text{high}}$ (indicating all *yes-no* questions are answered "*yes*"), we increment

Prompt P_{cap} for generating output caption

System Role: You are "Dual-Caption," an expert vision-language assistant. Your job is to describe (a) the user-supplied original image, and (b) the image that would exist after perfectly applying the user's editing instruction.

INPUT

The user will always provide both keys:

- `edit instruction`:
- `original image`:

OUTPUT (must follow exactly)

Return a single JSON object with two keys, in this order:

```
{"original_caption": "<1-2 sentences describing the original image>",
 "edited_caption": "<1-2 sentences describing the image after the edit>"}
```

STYLE & CONTENT RULES

- Base captions solely on visible content; no unfounded guesses.
- Mention dominant objects, attributes, actions, and setting.
- In `edited_caption`, describe the resulting scene only—do NOT mention the edit process or instruction.
  - (Bad: "The image is edited to…." Good: "A red balloon now floats above the dog….")
  - (Bad: "An apple, no pear…." Good: "An apple….")
- Each caption $\leq$ 40 English words.
- Do not quote the user's instruction verbatim; convey its visual effect instead.
- If the instruction is impossible or unsafe, produce the best policy-compliant visual outcome.

FINAL CHECKLIST

✓ Return exactly one JSON object—no extra text, blank lines, or comments.
✓ Keys must be `original_caption` and `edited_caption`, in that order.
✓ Follow length, tense, tone, and truthfulness requirements.

User Input:

`edit instruction`: ⟨edit_prompt⟩
`original image`: ⟨image⟩

Figure 8. **Prompt P_{cap} for generating output caption**.

Figure 9. **Redundancy appears in early preview images.** The figure displays clusters of visually similar images, grouped by colored boxes. We observe that candidates which are highly similar in the final output (bottom row) are also highly similar in their early previews (top row). This confirms that visual redundancy can be detected and filtered at an early stage.

the high-score count N_{cnt} (Lines 19-21). The image-score pair (I, S) is added to $\mathcal{U}$ (Line 22). ❺ The search terminates when $N_{\text{cnt}} = N_{\text{high}}$ (Lines 24-27), meaning sufficient intent-aligned results have been found. Finally, we return the set $\mathcal{U}$ (Line 29), and the candidate with the highest score is selected as the output.

B.3.1. Details of Retaining Top Results

Lines 10-23 of Alg. 6 implement the late-stage filter to retain the most promising candidates. Unlike the early pruning stage that uses a fixed rejection threshold S_{rj}, we adopt an adaptive filtering strategy at the late timestep t_l. This is motivated by the observation that preview scores at later denoising stages exhibit stronger correlation with final image quality. For each candidate, we generate a preview image $I_{0|t_l}$ and compute its unified score $S_{0|t_l}$ (Lines 6-9). We maintain a retain threshold S_{rt} that dynamically updates to the maximum score observed so far (Line 12). A candidate is retained if its score $S_{0|t_l}$ is within a tolerance δ of the current threshold: $S_{0|t_l} \geq S_{\text{rt}} - \delta$ (Line 11). This ensures that only candidates with scores comparable to the current best are fully denoised. This adaptive filter dynamically adjusts

Algorithm 6 `Adaptive_Stop` algorithm - Lines 5-6 of Alg. 3.

Require: source image I_{src}, text prompt c, intermediate latent set $\mathcal{X}_{t_e}$, prompt set $\mathcal{C}$, early step t_e and retain step t_l
Require: high-score number N_{high}, high-score threshold S_{high}, tolerance factor δ
1: $\mathcal{U} \leftarrow \{\}$ ▷ Initialize empty set for (I, S) pairs
2: $S_{\text{rt}} \leftarrow 0$ ▷ Initialize retain threshold
3: $N_{\text{cnt}} \leftarrow 0$ ▷ Initialize high-score count
4: ▷ **Depth-first generation**
5: **for** (x_{t_e}, c_r) in $(\mathcal{X}_{t_e}, \mathcal{C})$ **do**
6: $x_{t_l} \leftarrow \texttt{Sampler}(I_{\text{src}}, x_{t_e}, c_r, t_e, t_l)$ ▷ Sample from t_e to t_l
7: $x_{0|t_l} \leftarrow \texttt{One\_Step\_Preview}(x_{t_l}, t_l)$ ▷ Preview image from intermediate latent (*cf*. Sec 3.2 Eq. 4)
8: $I_{0|t_l} \leftarrow \texttt{VAE\_Decoder}(x_{0|t_l})$
9: $S_{0|t_l} \leftarrow \texttt{Vrf}(I_{\text{src}}, I_{0|t_l}, c)$ ▷ Compute unified score (*cf*. Sec 3.2, Eq. 8)
10: ▷ **Retain top results by evaluated score**
11: **if** $S_{0|t_l} >= S_{\text{rt}} - \delta$ **then**
12: $S_{\text{rt}} \leftarrow \max(S_{\text{rt}}, S_{0|t_l})$ ▷ Update retain threshold
13: $x_0 \leftarrow \texttt{Sampler}(I_{\text{src}}, x_{t_l}, c_r, t_l, 0)$ ▷ Sample from t_l to 0
14: $I \leftarrow \texttt{VAE\_Decoder}(x_0)$
15: $S \leftarrow \texttt{Vrf}(I_{\text{src}}, I, c)$ ▷ Compute unified score
16: ▷ **Instance-specific verification**
17: $S_{\text{spec}} \leftarrow \texttt{Specific\_Verifier}(I_{\text{src}}, I, c)$ ▷ Compute instance-specific score
18: $S \leftarrow S + S_{\text{spec}}$ ▷ Update evaluated score
19: **if** $S_{\text{spec}} \geq S_{\text{high}}$ **then**
20: $N_{\text{cnt}} \leftarrow N_{\text{cnt}} + 1$ ▷ Update high-score count
21: **end if**
22: $\mathcal{U} \leftarrow \mathcal{U} \cup \{(I, S)\}$
23: **end if**
24: ▷ **Stop when intent-aligned results suffice**
25: **if** $N_{\text{cnt}} == N_{\text{high}}$ **then**
26: break
27: **end if**
28: **end for**
29: **return** $\mathcal{U}$

to the quality distribution of candidates. It avoids wasting computation on samples that are unlikely to be optimal.

B.3.2. Details of Instance-Specific Verifier

The instance-specific verifier provides a fine-grained assessment to distinguish high-quality results from subtly flawed ones. It employs a two-stage process. First, using prompt P_q (shown in Fig. 10), the MLLM generates a set of five specific *yes/no* questions tailored to the edit instruction. These questions cover aspects such as instruction adherence and aesthetics. Second, using prompt P_a (shown in Fig. 11), the MLLM answers these questions for the fully generated image. As implemented in Alg. 6, the instance-specific score, S_{spec}, is calculated by the verifier (Line 17). This score is then added to the candidate's unified score to reward correctness (Line 18). We also track the number of intent-aligned candidates using a counter N_{cnt}, which is incremented if $S_{\text{spec}} \geq S_{\text{high}}$ (Lines 19–21). This counter is used to trigger the final opportunistic stopping condition.

C. Additional Experimental Results

C.1. Experimental Details

Our method, ADE-CoT, is built upon three open-sourced, state-of-the-art image editing models: Step1X-Edit [24], FLUX.1 Kontext [17], and BAGEL [8]. We adhere to their default configurations for the total number of denoising steps, which are $T = 28, 28, 50$, respectively. The early pruning timestep t_e and the late retaining timestep t_l are set based on the total steps for each model. The specific hyperparameter settings used in our experiments are detailed in Tab. 1. We use Qwen-VL-MAX [1] as the MLLM for all queries and VIE-Score [16] as our general verifier S_{gen}. To ensure the robustness of our findings, we conduct each scaling experiment three times with different random seeds for sampling noise. The results reported in the main paper are the average of these three runs. When multiple candidates achieve the same maximum score, we compute their pairwise visual similarity using DINOv2 [31] embeddings.

Prompt P_q for generating instance-specific questions

System Role: You are an expert AI assistant specializing in Image Editing Quality Assurance (QA). Your primary role is to act as a meticulous verifier of digital image edits. Your task is to analyze an Original Image and a corresponding Edit Instruction, and then generate a set of exactly 5 specific, verifiable questions. The fundamental rule is this: if a human reviewer answers "yes" to all 5 of your questions, it must unequivocally confirm that the edit was successfully and perfectly executed according to the instruction.

INPUT

- `Original Image`: [The initial image before editing]
- `Edit Instruction`: [A text description of the desired change]

OUTPUT (must follow exactly)

- Every question MUST be phrased so that a "yes" answer confirms a positive outcome.
- Return a single JSON object with the following structure:

```
{"questions": [
    "Question 1",
    "Question 2",
    "Question 3",
    "Question 4",
    "Question 5" ]}
```

CORE PRINCIPLES FOR QUESTION GENERATION

- **Instruction-Centric:** Every question must directly derive from the Edit Instruction. Deconstruct the instruction into its core components (e.g., object, action, style, location).
- **Binary & Objective:** Frame each question to be answerable with a simple and objective "Yes" or "No". Avoid subjective questions like "Does it look better?" and instead focus on verifiable facts, such as "Has the color of the car been changed from blue to red?".
- **Comprehensive Coverage:** Your 5 questions must collectively cover all aspects of the instruction. If the instruction is "Make the man taller and add a hat," you must have questions that verify both his height and the presence of the hat.
- **Negative Verification (No Collateral Damage):** At least one question must check for unintended side effects. This includes verifying that parts of the image not mentioned in the instruction remain unchanged, and that no new artifacts, blurs, or distortions have been introduced.
- **Holistic Quality Check:** At least one question must assess the overall integration and realism of the edit. It should check if the edit blends seamlessly with the rest of the image, maintaining consistent lighting, shadows, and texture.

User Input:

`Edit Instruction:` ⟨edit_instruction⟩
`Original Image:` ⟨image⟩

Figure 10. **Prompt P_q for generating instance-specific questions**.

We select the candidate with the highest average similarity to others as the centroid, representing the visual consensus among top-scoring outputs.

C.2. Details of Evaluation setting

Proposed efficiency metrics. We introduce two metrics to measure efficiency from different perspectives. ❶ **Reasoning Efficiency (η):** This metric is designed to measure the overall trade-off between performance and computational cost. An effective pruning strategy must not only reduce the Number of Function Evaluations (NFE) but also maintain high image quality. The design of η rewards methods that achieve high final scores with low NFE. The binary factor σ_i ensures that only methods achieving non-degraded performance are considered, which prevents strategies from gaining high efficiency scores by producing poor results. ❷ **Outcome Efficiency (ξ):** This metric is designed to quantify generation redundancy. Large-scale sampling in image editing often yields multiple correct outputs. An ideal method should find a satisfactory result quickly. ξ measures this by comparing the total NFE spent against the minimum NFE required to generate the first acceptable image.

Prompt P_a for answering instance-specific questions

System Role: You are an "Image-Edit Compliance Judge." For every dialogue turn you will receive:
- `EDIT_INSTRUCTION` – A text description of the desired change.
- `QUESTION_LIST` – exactly five yes/no questions, each asking whether a certain visual condition is true after the edit.
- `ORIGINAL_IMAGE` – The initial image before editing.
- `EDITED_IMAGE` – an edited version of the initial image.

YOUR TASK
- Imagine the edit is carried out exactly as written and reason about the resulting image.
- For each of the five questions decide "yes" (the condition is satisfied) or "no" (the condition is not satisfied or cannot be inferred). When unsure, answer "no."
- Return nothing except a JSON object with five keys: `"Q1"`, `"Q2"`, `"Q3"`, `"Q4"`, `"Q5"`.
- The value of every key must be the lowercase string "yes" or "no".
- Do not output any explanations, comments, or additional keys.

OUTPUT (must follow exactly)
Return a single JSON object with the following structure:

```
{"Q1": "yes|no",
 "Q2": "yes|no",
 "Q3": "yes|no",
 "Q4": "yes|no",
 "Q5": "yes|no"}
```

User Input:
`EDIT_INSTRUCTION:` ⟨edit_instruction⟩
`QUESTION_LIST:` ⟨instance-specific_question⟩
`ORIGINAL_IMAGE:` ⟨original_image⟩
`EDITED_IMAGE:` ⟨edited_image⟩

Figure 11. **Prompt P_a for answering instance-specific questions**.

Table 1. **Default hyperparameters used in our experiments.**

Hyper.	Value	Description
T	28, 28, 50	Total denoising steps for each model.
t_e	8, 8, 16	Timestep for early preview and pruning.
t_l	16, 16, 36	Timestep for late retaining.
N_{min}	1	Minimum sampling budget.
γ	0.15	Sensitivity for difficulty-aware allocation.
S_{max}	10	Maximum score for normalization.
λ_{reg}	1	Weight for the region correctness score.
λ_{cap}	3	Weight for the caption consistency score.
S_{rj}	5	Rejection threshold for early pruning.
τ_{sim}	0.98	Similarity threshold for filtering candidates.
N_{high}	4	Number of intent-aligned results to stop.

A higher ξ indicates that the method wastes less computation on producing redundant correct images.

Two comparison settings. We analyze the performance of all methods from two different settings to provide a comprehensive comparison. ❶ **Results under fixed sampling budget.** In this setting, all methods start with the same initial sampling budget (e.g., $N = 32$). This comparison aims to identify which method achieves the best performance and efficiency when allocated a fixed amount of computational resources. It directly shows the effectiveness of different pruning and search strategies. ❷ **Results under comparable performance.** In this setting, we compare the computational cost of methods that achieve a similar quality level, specifically a non-degraded performance relative to the Best-of-N (BoN) baseline. The goal is to measure the actual speedup a method provides while maintaining a target performance. This highlights the practical value of a strategy in terms of saving time and resources.

C.3. More Ablation Studies

We present a comprehensive ablation study of our key components in Tab. 2. Most of these results have been analyzed in the main paper. Here, we provide additional analyses to further validate the effectiveness of our design choices.

Are both edited-region correctness and instruction-caption consistency effective? To isolate the impact of

Table 2. **Effect of the three proposed strategies on efficiency and performance**. We evaluate our method on GEdit-Bench [24].

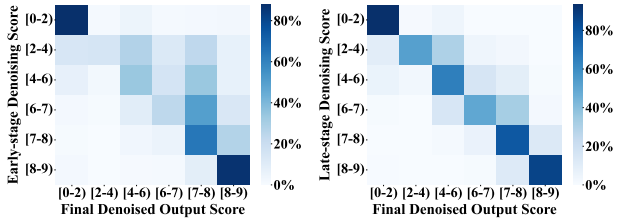
Model	N	FLUX.1 Kontext [17]				BAGEL [8]				Step1X-Edit [24]			
		G_O ↑	NFE ↓	η ↑	ξ ↑	G_O ↑	NFE ↓	η ↑	ξ ↑	G_O ↑	NFE ↓	η ↑	ξ ↑
Baseline (BoN)	32	6.641	896	0.66	0.11	6.908	1600	0.69	0.14	7.157	896	0.72	0.13
a) + adaptive sampling	32	6.641	797	0.74	0.26	6.909	1391	0.76	0.23	7.157	778	0.81	0.27
b) + early filter by general verifier	32	6.642	719	0.81	0.40	6.912	1351	0.79	0.39	7.157	719	0.89	0.42
c) + early filter (+ S_{reg})	32	6.645	687	0.84	0.42	6.915	1321	0.84	0.43	7.158	678	0.95	0.45
d) + early filter (+ S_{cap})	32	6.647	673	0.87	0.44	6.916	1290	0.88	0.45	7.161	638	1.02	0.47
e) + removing visually similar samples	32	6.651	508	1.26	0.58	6.915	1087	1.02	0.50	7.162	522	1.22	0.54
f) + retaining top results at late stage	32	6.652	464	1.34	0.61	6.935	972	1.08	0.54	7.163	462	1.34	0.58
g) + instance-specific verifier	32	**6.702**	464	1.37	0.63	**6.984**	972	1.12	0.57	**7.206**	462	1.36	0.60
h) + opportunistic stopping (**full model**)	32	6.695	**418**	**1.47**	**0.66**	6.972	**882**	**1.27**	**0.62**	7.196	**434**	**1.45**	**0.62**

our edit-specific verifiers, we start from a baseline that uses only early filtering with a general verifier (S_{gen}) (row b). As shown in Tab. 2, adding the edited-region correctness score (S_{reg}) (row c) consistently reduces NFE across all models while maintaining performance. This indicates that S_{reg} is effective at pruning candidates with incorrect edit localization. We then further incorporate the instruction-caption consistency score (S_{cap}) (row d). This step yields additional efficiency gains, reducing the NFE for Step1X-Edit from 678 to 638 and for FLUX.1 Kontext from 687 to 673. These results confirm that both S_{reg} and S_{cap} are effective and complementary. They target distinct failure modes—localization and semantic alignment, respectively—and their combined use significantly enhances the precision of our early pruning strategy.

Why fixed thresholds for early pruning but dynamic thresholds for late retaining? Our choice of pruning thresholds is based on the correlation between intermediate and final scores at different denoising stages, as shown in Fig. 12. In the early stage, the correlation between preview scores and final scores is moderate. A low preview score does not guarantee a low final score. An aggressive pruning strategy is therefore risky, as it could discard high-potential candidates. We use a fixed, conservative threshold to safely remove only clear failures. Conversely, the correlation becomes much stronger in the late stage. Fig. 12(b) shows that late-stage scores are highly predictive of the final quality. This strong correlation allows for a more aggressive strategy. We apply a dynamic threshold to filter out candidates that are unlikely to surpass the current best, which improves efficiency without performance loss.

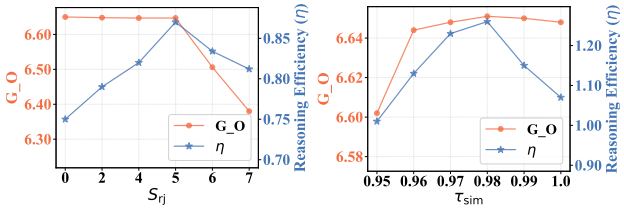
C.4. More Hyperparameter Analysis

What are the optimal hyperparameters S_{rj} and τ_{sim}? We analyze the impact of the rejection threshold S_{rj} and the similarity threshold τ_{sim} in Fig. 13. Increasing S_{rj} improves reasoning efficiency by pruning low-potential samples, while performance (G_O) remains stable up to a threshold of 5. However, a higher S_{rj} may remove potentially correct candidates, causing both metrics to decline.



(a) Early denoising stage. (b) Late denoising stage.

Figure 12. **Correlation between preview scores and final performance across denoising stages.** We show correlation matrixs between intermediate scores (y-axis) and final scores (x-axis) at **(a)** the early and **(b)** late denoising stages.



(a) S_{rj} (b) τ_{sim}

Figure 13. **More hyperparameter analysis**.

For τ_{sim}, decreasing it removes redundant images, which improves both performance and reasoning efficiency, peaking at $\tau_{sim} = 0.98$. This may be because discarded similar images also tend to have low potential scores. A further decrease causes a sharp drop in both metrics. Thus, we set $S_{rj} = 5$ and $\tau_{sim} = 0.98$ as our default values.

C.5. More Analysis of Cost Computation

Beyond the Number of Function Evaluations (NFE), we analyze the computational cost of MLLM queries in our framework. Tab. 3 reports the average number of MLLM calls per editing case on GEdit-Bench under a sampling budget of $N = 32$ across three models. We observe that the general MLLM verifier, which is also required by all prior Image-CoT methods [26, 57, 59], accounts for the majority of queries in our framework. This component evaluates multiple candidates during early pruning and late retaining stages. In contrast, verifiers introduced by our method, including edited-region localization (via prompt

Table 3. **Average MLLM queries per editing case on GEdit-Bench**. We report the average number of calls for each MLLM component under a sampling budget of $N = 32$.

MLLM Component	Kontext	BAGEL	Step1X-Edit
General Verifier	65.1	71.4	66.1
Region Generation	1.0	1.0	1.0
Caption Generation	1.0	1.0	1.0
Instance-Specific Questions (Generate)	1.0	1.0	1.0
Instance-Specific Questions (Answer)	14.9	17.6	15.5
Total Avg.	**83.0**	**92.0**	**84.6**

P_{reg}), instruction-caption consistency (via prompt P_{cap}), and the instance-specific verifier, contribute minimal overhead. Specifically, region and caption generation require only 1.0 query per case. The instance-specific verifier generates questions once per case and answers them for a small number of top candidates, resulting in an average of 14.9, 17.6, and 15.5 queries for Kontext, BAGEL, and Step1X-Edit, respectively. The total average MLLM calls per case range from 83.0 to 92.0 across different models. This demonstrates that our edit-specific verification strategies introduce limited additional MLLM overhead while achieving significant NFE reduction compared to baseline methods.

C.6. More Qualitative Results

We present additional qualitative results to demonstrate the effectiveness of ADE-CoT across different editing scenarios. Fig. 1 and Fig. 2 show that our ADE-CoT significantly improves baseline model performance on complex edits and multi-turn edits. In Fig. 1, baseline models often fail on challenging tasks such as large pose changes, multi-object modifications, and fine-grained regional edits. Our method successfully handles these cases through adaptive test-time scaling. Fig. 2 demonstrates that baseline models struggle to maintain consistency across multiple sequential instructions, leading to accumulated errors in subsequent turns. In contrast, ADE-CoT preserves context from previous edits and produces correct final images that reflect all requested changes. Fig. 14 illustrates the advantage of our instance-specific verifier in final selection. Both baseline and Best-of-N methods fail to detect subtle errors in edited results. Our ADE-CoT generates targeted questions to examine critical details, enabling more accurate identification of correct outputs and producing superior final results compared to Best-of-N selection.

C.7. Critical Analysis and Discussion

How much do MLLMs impact our framework? Our framework, ADE-CoT, relies on MLLMs for key verification steps. It uses MLLMs to identify the edit region, generate a target caption, and compute the general score. Due to hallucinations, these models inevitably introduce incorrect judgments. To verify the impact of MLLM capability on our framework, we conduct experiments with three different MLLMs. We find that ADE-CoT demonstrates strong robustness across varying MLLM capacities. ❶ As shown in Tab. 5 of the main paper, our method consistently achieves over 2× speedup compared to BoN across all tested MLLMs. ❷ We observe that even when using a weaker MLLM such as Qwen2.5-VL-72B, replacing specific components with stronger MLLMs leads to consistent improvements. Tab. 4 shows that replacing the MLLM for region localization improves edited-region correctness and overall performance. Similarly, Tab. 5 demonstrates that using a stronger MLLM for caption generation enhances instruction-caption consistency. These results show that more accurate predictions from better MLLMs lead to consistent gains in both performance and efficiency.

Table 4. **Effect of MLLMs on edited-region correctness**. We compare three MLLMs for edited-region localization (via prompt P_{reg}), while keeping Qwen-VL-72B for other components.

Model	MLLM for Region	Kontext		BAGEL		Step1X-Edit	
		G_O↑	NFE↓	G_O↑	NFE↓	G_O↑	NFE↓
BoN	-	6.641	896	6.908	1600	7.157	896
ADE-CoT	Qwen2.5-VL-72B [1]	6.637	436	7.042	897	7.193	446
	Qwen-VL-MAX [1]	6.661	428	6.993	901	7.194	445
	Qwen3-VL-32B [53]	**6.673**	**424**	**7.048**	**869**	**7.198**	**436**

Table 5. **Effect of MLLMs on instruction-caption consistency**. We compare three MLLMs for instruction-caption consistency (via prompt P_{cap}), while keeping Qwen-VL-72B for others.

Model	MLLM for Caption	Kontext		BAGEL		Step1X-Edit	
		G_O↑	NFE↓	G_O↑	NFE↓	G_O↑	NFE↓
BoN	-	6.641	896	6.908	1600	7.157	896
ADE-CoT	Qwen2.5-VL-72B [1]	6.637	436	7.042	897	7.193	446
	Qwen-VL-MAX [1]	6.651	**419**	7.021	899	7.186	450
	Qwen3-VL-32B [53]	**6.664**	423	**7.052**	**883**	**7.195**	**440**

Can Image-CoT improve all editing cases? As illustrated in Fig. 5, we find that some samples still receive low scores even after applying Image-CoT. These cases typically represent scenarios where the model inherently lacks editing capability. Even after extensive sampling or prompt modification, their performance shows minimal improvement. It also demonstrates that Image-CoT can serve as a diagnostic method to identify model capability boundaries. By incorporating these cases into training data, we can further enhance the capabilities of existing models.

D. Limitations and Future Work

Limitations. Despite achieving superior trade-offs between performance and efficiency, our method suffers from the following limitations: ❶ **MLLM computational overhead**. Our verification relies on large-scale MLLMs (*e.g.*, Qwen-VL 72B), which increases inference latency and limits in resource-constrained scenarios. ❷ **Hallucination in**

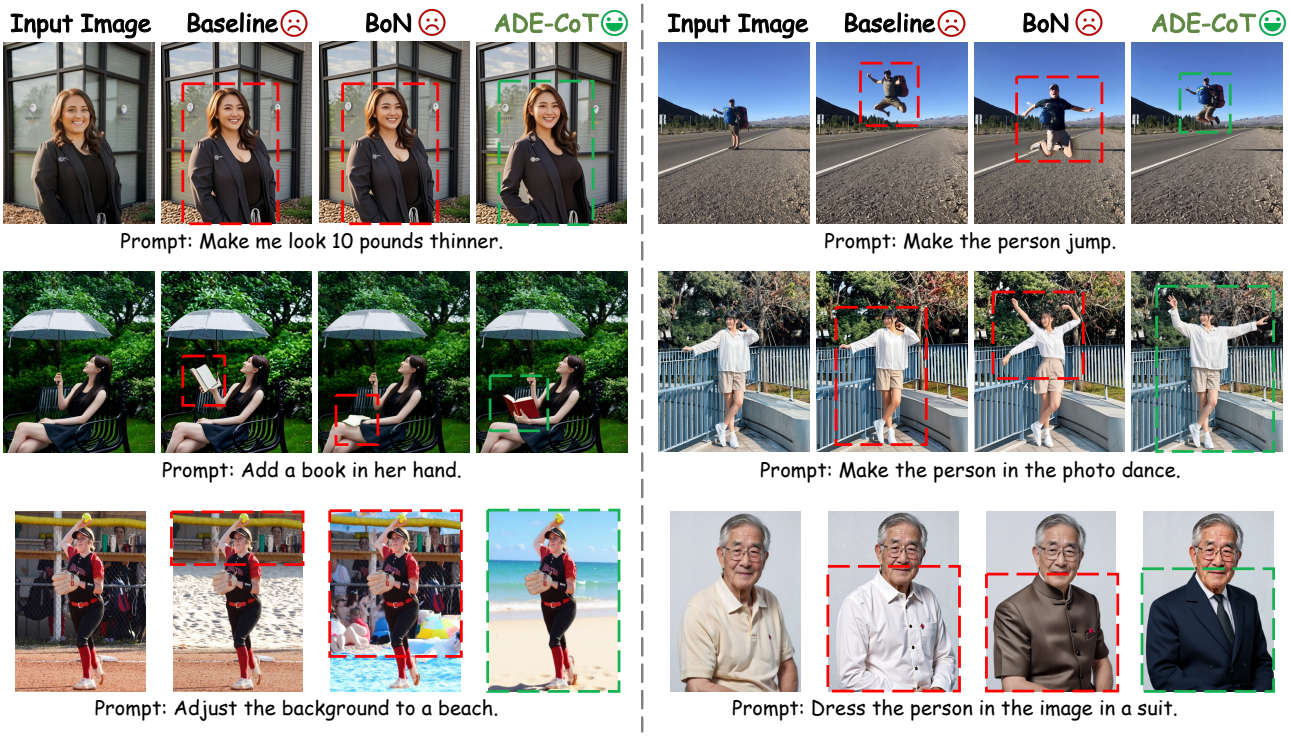


Figure 14. **Instance-specific verifier improves fine-grained selection**. We compare the baseline model, Best-of-N (BoN), and our ADE-CoT across challenging editing scenarios. Baseline models and BoN often produce results with subtle errors (marked in red boxes), such as incorrect pose changes, background adjustments, and clothing modifications. In contrast, our instance-specific verifier generates targeted questions to examine critical details, enabling accurate detection of editing errors and selecting correct results (marked in green boxes).

verification. MLLMs may generate hallucinations during the verification process. This can compromise the accuracy of generated captions, region masks, and instance-specific questions, leading to incorrect quality assessments. While our experiments demonstrate that different MLLMs show minimal variance when ranking multiple candidates (Sec. 4.3, Tab. 5), accurately determining whether the edited image fully satisfies user intent remains challenging.

Future work. We identify two primary directions for future research based on these limitations. ❶ **Efficient and accurate verification models**. A key direction is to utilize smaller, specialized models for evaluation. Lightweight models (*e.g.*, 7B parameters) could be trained to provide fast and accurate assessments of edited images. Additionally, an accurate evaluation model would be particularly effective for evaluating intermediate preview images during the denoising process. This could enhance the proposed edit-specific verification and opportunistic stopping strategies, further improving overall efficiency. ❷ **Broader applications**. Our ADE-CoT framework could be extended to other goal-directed generation tasks. 1 Its core strategies of difficulty-aware resource allocation and opportunistic stopping are applicable to domains like video editing and multi-turn conversational generation. These strategies may also be beneficial for standard text-to-image and text-to-video generation, enabling a more efficient Image-CoT process.

E. Extended Related Work

Test-time scaling in image generation. Recent years have seen rapid development in generative models [4, 8, 14, 17–19, 24, 28, 41, 42, 44–47, 49, 52]. Test-time scaling, as a training-free approach, aims to improve generation quality by extending the inference time. Early work on diffusion-based models investigates this by scaling the number of denoising steps [15, 25, 36, 38–40]. More recently, following the success of Chain-of-Thought (CoT) [2, 5–7, 20, 22, 27, 29, 33, 34, 48, 54–56], Image-CoT has emerged as a promising paradigm. The standard Image-CoT method is noise scaling [26], which generates multiple candidates by perturbing the noise and selects the best image as the final output. While effective, its computational cost scales linearly with the number of candidates. Subsequent work aims to generate higher-quality candidates within a fixed budget. Some methods enhance candidate diversity through prompt-level intervention, which enhances prompts by rewriting [13, 30] or reflective updates for iterative refinement [21, 57, 60]. Another line of work adapts search algorithms to the diffusion process. These methods treat the reverse diffusion chain as a search trajec-

tory [9, 12, 26, 37, 58]. They change the noise based on verifier scores to select promising sampling directions. Recent methods [23, 32, 43, 50, 51, 57, 59] utilize MLLMs as a verifier to prune low-potential trajectories at early denoising stages. However, most Image-CoT methods focus on text-to-image generation. They are inefficient for image editing due to three key issues: (1) fixed sampling budgets waste computation on simple edits, (2) general MLLM scores cause misjudgement during early pruning, and (3) large-scale sampling produces redundant correct outputs. To address these challenges, we propose ADE-CoT, an edit-specific test-time scaling method that incorporates difficulty-aware resource allocation, edit-specific verification, and depth-first opportunistic stopping.

Verifiers for image editing can be divided into two primary approaches. The first approach comprises metrics requiring ground-truth edited images and corresponding output captions. These methods typically utilize CLIP Score [10] for image-text alignment, CLIP [35] and DINO [3] similarity for image-image comparison, and L1/L2 distance for pixel-level similarity. However, such metrics are difficult to apply in test-time scaling (TTS) scenarios, as ground-truth data are unavailable during inference. The second approach leverages MLLMs as verifiers. Existing methods, such as VIEScore [16] and HQScore [11], use general verifiers that use instance-agnostic prompts to evaluate aesthetic quality and semantic consistency for each instance. However, general verifiers face two limitations in editing Image-CoT. In early denoising stages, they may incorrectly pruning high-potential candidates that exhibit low preview quality. In the final selection, they struggle to distinguish subtle differences between candidates, assigning identical high scores to images with minor errors. To address these limitations, we introduce two complementary verification strategies. For early-stage misjudgement, we propose edit-specific verification that queries MLLMs to generate ground-truth captions and expected edit regions, enabling more accurate assessment of caption consistency and region localization. For final-stage selection, we introduce instance-specific verification that first generates targeted *yes-no* questions about specific editing aspects, then provides answers based on these questions. This two-stage inquiry guides MLLMs to notice critical details and provides the decision for opportunistic stopping, improving fine-grained selection accuracy, and reducing redundant outputs.

References

[1] Shuai Bai, Keqin Chen, and Xuejing Liu et al. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923, 2025. 13, 17

[2] Sule Bai, Mingxing Li, Yong Liu, Jing Tang, Haoji Zhang, Lei Sun, Xiangxiang Chu, and Yansong Tang. Univg-r1: Reasoning guided universal visual grounding with reinforcement learning. *arXiv preprint arXiv:2505.14231*, 2025. 18

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9630–9640, 2021. 19

[4] Chubin Chen, Jiashu Zhu, Xiaokun Feng, Nisha Huang, Meiqi Wu, Fangyuan Mao, Jiahong Wu, Xiangxiang Chu, and Xiu Li. S^2-Guidance: Stochastic Self Guidance for Training-Free Enhancement of Diffusion Models. *arXiv preprint arXiv:2508.12880*, 2025. 18

[5] Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and Sercan Ö. Arik. SETS: leveraging self-verification and self-correction for improved test-time scaling. *CoRR*, abs/2501.19306, 2025. 18

[6] Rui Chen, Lei Sun, Jing Tang, Geng Li, and Xiangxiang Chu. Finger: Content aware fine-grained evaluation with reasoning for ai-generated videos. In *ACM MM*, pages 3517–3526, 2025.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. 18

[8] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Shi Guang, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *CoRR*, abs/2505.14683, 2025. 1, 3, 13, 16, 18

[9] Haoran He, Jiajun Liang, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, and Ling Pan. Scaling image and video generation via test-time evolutionary search. *CoRR*, abs/2505.17618, 2025. 19

[10] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *EMNLP*, pages 7514–7528, 2021. 19

[11] Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Cihang Xie, and Yuyin Zhou. Hq-edit: A high-quality dataset for instruction-based image editing. In *ICLR*, 2025. 19

[12] Vineet Jain, Kusha Sareen, Mohammad Pedramfar, and Siamak Ravanbakhsh. Diffusion tree sampling: Scalable inference-time alignment of diffusion models. *CoRR*, abs/2506.20701, 2025. 19

[13] Dongzhi Jiang, Ziyu Guo, Renrui Zhang, Zhuofan Zong, Hao Li, Le Zhuo, Shilin Yan, Pheng-Ann Heng, and Hongsheng Li. T2I-R1: reinforcing image generation with collaborative semantic-level and token-level cot. *CoRR*, abs/2505.00703, 2025. 18

[14] Dongyang Jin, Ryan Xu, Jianhao Zeng, Rui Lan, Yancheng Bai, Lei Sun, and Xiangxiang Chu. Semantic context

matters: Improving conditioning for autoregressive models, 2026. 18

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 18

[16] Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. Viescore: Towards explainable metrics for conditional image synthesis evaluation. In *ACL*, pages 12268–12290, 2024. 5, 9, 13, 19

[17] 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. *CoRR*, abs/2506.15742, 2025. 1, 3, 13, 16, 18

[18] Rui Lan, Yancheng Bai, Xu Duan, Mingxing Li, Dongyang Jin, Ryan Xu, Lei Sun, and Xiangxiang Chu. Flux-text: A simple and advanced diffusion transformer baseline for scene text editing. *arXiv preprint arXiv:2505.03329*, 2025.

[19] Huaqiu Li, Yong Wang, Tongwen Huang, Hailang Huang, Haoqian Wang, and Xiangxiang Chu. Ld-rps: Zero-shot unified image restoration via latent diffusion recurrent posterior sampling. *arXiv preprint arXiv:2507.00790*, 2025. 18

[20] Mingxing Li, Rui Wang, Lei Sun, Yancheng Bai, and Xiangxiang Chu. Next token is enough: Realistic image quality and aesthetic scoring with multimodal large language model. *arXiv preprint arXiv:2503.06141*, 2025. 18

[21] Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Arsh Koneru, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Reflect-dit: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. *CoRR*, abs/2503.12271, 2025. 2, 18

[22] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. 18

[23] Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-t1: Test-time scaling for video generation. *CoRR*, abs/2503.18942, 2025. 2, 19

[24] Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025. 1, 3, 13, 16, 18

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Mach. Intell. Res.*, 22(4):730–751, 2025. 18

[26] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi S. Jaakkola, Xuhui Jia, and Saining Xie. Scaling inference time compute for diffusion models. In *CVPR*, pages 2523–2534, 2025. 2, 6, 16, 18, 19

[27] Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning. *CoRR*, abs/2310.10080, 2023. 18

[28] Fangyuan Mao, Aiming Hao, Jintao Chen, Dongxia Liu, Xiaokun Feng, Jiashu Zhu, Meiqi Wu, Chubin Chen, Jiahong Wu, and Xiangxiang Chu. Omni-effects: Unified and spatially-controllable visual effects generation. *arXiv preprint arXiv:2508.07981*, 2025. 18

[29] Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. *CoRR*, abs/2503.04412, 2025. 18

[30] Yoonjin Oh, Yongjin Kim, Hyomin Kim, Donghwan Chi, and Sungwoong Kim. Object-centric self-improving preference optimization for text-to-image generation. *CoRR*, abs/2506.02015, 2025. 18

[31] Maxime Oquab, Timothée Darcet, and Théo Moutakanni et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024, 2024. 13

[32] Yuta Oshima, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Inference-time text-to-video alignment with diffusion latent beam search. *CoRR*, abs/2501.19252, 2025. 19

[33] Isha Puri, Shivchander Sudalairaj, Guangxuan Xu, Kai Xu, and Akash Srivastava. A probabilistic inference approach to inference-time scaling of llms using particle-based monte carlo methods. *CoRR*, abs/2502.01618, 2025. 18

[34] Xiangyan Qu, Gaopeng Gou, Jiamin Zhuang, Jing Yu, Kun Song, Qihao Wang, Yili Li, and Gang Xiong. Proapo: Progressively automatic prompt optimization for visual classification. In *CVPR*, pages 25145–25155, 2025. 18

[35] Alec Radford, Jong Wook Kim, and Chris Hallacy et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 19

[36] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 18

[37] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *CoRR*, abs/2501.06848, 2025. 19

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 18

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*. OpenReview.net, 2021.

[40] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, pages 32211–32252, 2023. 18

[41] Datao Tang, Xiangyong Cao, Xingsong Hou, Zhongyuan Jiang, Junmin Liu, and Deyu Meng. Crs-diff: Controllable remote sensing image generation with diffusion model. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 18

[42] Datao Tang, Xiangyong Cao, Xuan Wu, Jialin Li, Jing Yao, Xueru Bai, Dongsheng Jiang, Yin Li, and Deyu Meng. Aerogen: Enhancing remote sensing object detection with diffusion-driven data generation. In *Proceedings of the*

Computer Vision and Pattern Recognition Conference, pages 3614–3624, 2025. 18

[43] Rui Tian, Mingfei Gao, Mingze Xu, Jiaming Hu, Jiasen Lu, Zuxuan Wu, Yinfei Yang, and Afshin Dehghan. Unigen: Enhanced training & test-time strategies for unified multimodal understanding and generation. *CoRR*, abs/2505.14682, 2025. 19

[44] Jiyuan Wang, Chunyu Lin, Lang Nie, Kang Liao, Shuwei Shao, and Yao Zhao. Digging into contrastive learning for robust depth estimation with diffusion models. In *ACM MM*, page 4129–4137, 2024. 18

[45] Jiyuan Wang, Chunyu Lin, Cheng Guan, Lang Nie, Jing He, Haodong Li, Kang Liao, and Yao Zhao. Jasmine: Harnessing diffusion prior for self-supervised depth estimation, 2025.

[46] JiYuan Wang, Chunyu Lin, Lei Sun, Rongying Liu, Lang Nie, Mingxing Li, Kang Liao, Xiangxiang Chu, and Yao Zhao. From editor to dense geometry estimator, 2025.

[47] Jiyuan Wang, Chunyu Lin, Lei Sun, Zhi Cao, Yuyang Yin, Lang Nie, Zhenlong Yuan, Xiangxiang Chu, Yunchao Wei, Kang Liao, and Guosheng Lin. Geometry-guided reinforcement learning for multi-view consistent 3d scene editing, 2026. 18

[48] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022. 18

[49] Chenfei Wu, Jiahao Li, and Jingren Zhou et al. Qwen-image technical report. *CoRR*, abs/2508.02324, 2025. 1, 18

[50] Meiqi Wu, Jiashu Zhu, Xiaokun Feng, Chubin Chen, Chen Zhu, Bingze Song, Fangyuan Mao, Jiahong Wu, Xiangxiang Chu, and Kaiqi Huang. Imagerysearch: Adaptive test-time search for video generation beyond semantic dependency constraints. *arXiv preprint arXiv:2510.14847*, 2025. 19

[51] Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, Bingchen Liu, Daquan Zhou, and Song Han. SANA 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *CoRR*, abs/2501.18427, 2025. 2, 19

[52] Ryan Xu, Dongyang Jin, Yancheng Bai, Rui Lan, Xu Duan, Lei Sun, and Xiangxiang Chu. Scalar: Scale-wise controllable visual autoregressive learning. *arXiv preprint arXiv:2507.19946*, 2025. 18

[53] An Yang, Anfeng Li, and Baosong Yang et al. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. 17

[54] Zhenlong Yuan, Xiangyan Qu, Chengxuan Qian, Rui Chen, Jing Tang, Lei Sun, Xiangxiang Chu, Dapeng Zhang, Yiwei Wang, Yujun Cai, and Shuo Li. Video-star: Reinforcing open-vocabulary action recognition with tools. *arXiv preprint arXiv:2510.08480*, 2025. 18

[55] Zhenlong Yuan, Jing Tang, Jinguo Luo, Rui Chen, Chengxuan Qian, Lei Sun, Xiangxiang Chu, Yujun Cai, Dapeng Zhang, and Shuo Li. Autodrive-r2: Incentivizing reasoning and self-reflection capacity for vla model in autonomous driving. *arXiv preprint arXiv:2509.01944*, 2025.

[56] Zhenlong Yuan, Xiangyan Qu, Jing Tang, Rui Chen, Lei Sun, Ruidong Chen, Hongwei Yu, Chengxuan Qian, Xiangxiang Chu, Shuo Li, and Yuyin Zhou. What if agents could imagine? reinforcing open-vocabulary hoi comprehension through generation, 2026. 18

[57] Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Ziyu Guo, Haoquan Zhang, Manyuan Zhang, Jiaming Liu, Peng Gao, and Hongsheng Li. Let's verify and reinforce image generation step by step. In *CVPR*, pages 28662–28672, 2025. 2, 5, 6, 16, 18, 19

[58] Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Y. Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. *CoRR*, abs/2505.23614, 2025. 19

[59] Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. *CoRR*, abs/2504.20690, 2025. 2, 5, 6, 9, 16, 19

[60] Le Zhuo, Liangbing Zhao, Sayak Paul, Yue Liao, Renrui Zhang, Yi Xin, Peng Gao, Mohamed Elhoseiny, and Hongsheng Li. From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. *CoRR*, abs/2504.16080, 2025. 18