RAISE: Requirement-Adaptive Evolutionary Refinement for Training-Free Text-to-Image Alignment

Supplementary Material

Table 4. **Evaluation with different base DMs (FLUX.1-dev [19], FLUX.1-schnell [19], SANA-1.5 4.8B [47]) on GenEval [11].** The best and second best results are **bolded** and underlined. "Avg. #Samples Generated" and "Avg. #Calls VLM" indicate efficiency.

Methods	Diffusion Model's Steps / Size	Avg. #Samples Generated	Avg. #Calls VLM	GenEval Score						
				Overall	Single Object	Two Object	Counting	Colors	Position	Attribute Binding
FLUX.1-dev [19]	28 / 12B	1	0	0.67	0.99	0.81	0.75	0.80	0.21	0.48
+ ❄ T2I-Copilot [5]	28 / 12B	11.3	22.6	0.74	0.99	0.91	0.68	0.86	0.55	0.46
+ 🔥 ReflectionFlow [51]	28 / 12B	32	64	0.91	**1.00**	0.98	0.89	0.95	0.89	0.75
+ ❄ RAISE (Ours)	28 / 12B	18.6	7.3	**0.94**	**1.00**	**1.00**	**0.95**	**0.98**	0.83	**0.87**
FLUX.1-schnell [19]	4 / 12B	1	0	0.66	**1.00**	0.87	0.59	0.76	0.29	0.45
+ ❄ RAISE (Ours)	4 / 12B	18.7	7.3	0.93	**1.00**	**1.00**	0.93	**0.98**	0.83	0.85
SANA-1.5 4.8B [47]	20 / 4.8B	1	0	0.72	0.99	0.85	0.77	0.87	0.34	0.54
+ ❄ RAISE (Ours)	20 / 4.8B	19.4	7.7	0.93	**1.00**	**1.00**	0.91	0.96	**0.92**	0.81

Table 5. **Evaluation of RAISE with different base VLMs [1, 12, 39] on GenEval [11].** The best and second best results are **bolded** and underlined. "Avg. #Samples Generated" and "Avg. #Calls VLM" indicate efficiency. 🔒 denotes proprietary models, 🔥 denotes fine-tuned open-source models, and ❄ denotes frozen open-source models that do not require any additional fine-tuning.

Methods	Avg. #Samples Generated	Avg. #Calls VLM	GenEval Score						
			Overall	Single Object	Two Object	Counting	Colors	Position	Attribute Binding
FLUX.1-dev [19]	1	0	0.67	0.99	0.81	0.75	0.80	0.21	0.48
T2I-Copilot [5] (❄ FLUX.1-dev [19] + ❄ Mistral-Small-3.2-24B [1])	11.3	22.6	0.74	0.99	0.91	0.68	0.86	0.55	0.46
ReflectionFlow [51] (🔥 FLUX.1-dev [19] + 🔒 GPT-4o [15] + 🔥 Qwen2.5-VL-7B [2])	32	64	0.91	**1.00**	0.98	0.89	0.95	**0.89**	0.75
RAISE (❄ FLUX.1-dev [19] + ❄ gemma-3-27b-it [12])	18.4	7.2	0.92	**1.00**	0.99	0.93	0.97	0.78	0.86
RAISE (❄ FLUX.1-dev [19] + ❄ Qwen3-VL-32B-Instruct [39])	20.3	8.1	0.93	**1.00**	**1.00**	0.91	0.94	**0.89**	0.83
RAISE (❄ FLUX.1-dev [19] + ❄ Mistral-Small-3.2-24B [1])	18.6	7.3	**0.94**	**1.00**	**1.00**	**0.95**	**0.98**	0.83	**0.87**

6. Additional Results

6.1. RAISE with Different Base Models

RAISE is a training-free and plug-and-play framework that requires no additional tuning of either the diffusion model (DM) or the vision–language model (VLM), allowing it to be seamlessly applied to new DM base models and new VLM base models. As shown in this section, RAISE is model-agnostic and delivers consistent gains across both types of base models, demonstrating efficient and generalizable multi-round self-improvement.

Different Base Diffusion Models (DMs). Table 4 reports the results of RAISE when paired with a range of DMs. Across all tested DMs, which vary in model size, number of diffusion steps, and generation quality, RAISE consistently improves prompt–image alignment and achieves high overall GenEval scores of 0.93–0.94. These results indicate that RAISE is robust to the choice of diffusion model and operates in a plug-and-play manner without any DM fine-tuning.

Different Base Vision-Language Models (VLMs). Table 5 summarizes the performance of RAISE when combined with different VLM backbones. Across all tested VLMs, which vary in family and parameter size, RAISE delivers consistent improvements in alignment and strong overall GenEval scores of 0.92-0.94. This shows that RAISE adapts effectively to different VLM reasoning capabilities, despite differences in architecture, scale, and training strategy. It requires no VLM fine-tuning or proprietary models, highlighting its model-agnostic plug-and-play nature.

Table 6. **Efficiency comparison on GenEval [11].** RAISE consistently achieves the highest GenEval score across budgets (Max #Samples = 8, 16, 32). At 32 samples, it requires *41.9% fewer samples generated* and *88.6% fewer VLM calls* on average than the second-best method.

Methods	**GenEval Score**	Max. #Samples Allowed	**Avg. #Samples Generated**	Max. #Calls VLM	**Avg. #Calls VLM**
FLUX.1-dev [19]	0.67	1	1	0	0
+ ❄ T2I-Copilot [5]	0.75	8	**4.3**	17	8.6
+ 🔥 ReflectionFlow [51]	0.86	8	8	16	16
+ ❄ **RAISE** (1 round)	**0.89**	8	8	3	**3**
+ ❄ T2I-Copilot [5]	0.75	16	**6.9**	33	13.8
+ 🔥 ReflectionFlow [51]	0.90	16	16	32	32
+ ❄ **RAISE** (2 rounds)	**0.92**	16	16	6	**6**
+ ❄ T2I-Copilot [5]	0.74	32	**11.3**	65	22.6
+ 🔥 ReflectionFlow [51]	0.91	32	32	64	64
+ ❄ **RAISE** ($\leq$ 4 rounds)	**0.94**	32	18.6	14	**7.3**

6.2. Details on Efficiency and Adaptive-Scaling

As shown in Table 6 and Fig. 4, RAISE continues to improve as the sampling budget increases (max number of samples allowed: 8, 16, 32), achieving the highest GenEval [11] scores across all budget settings. RAISE achieves superior alignment while achieving better efficiency by dynamically allocating computation. For example, with a max budget of 32 samples, it generates 41.9% fewer samples (18.6 vs. 32) and requires 88.6% fewer VLM calls (7.3 vs. 64) than training-based inference-time scaling approaches on GenEval [11], while requiring no additional model training.

In contrast to reflection-tuned and other training-free inference-time scaling methods that plateau early or fail to improve with additional samples, RAISE maintains a strong performance–efficiency Pareto frontier and achieves steady gains through effective use of additional computation, highlighting the scalability of its requirement-driven, evolutionary multi-action refinement design.

The efficiency of RAISE stems from its requirement-adaptive scaling, directing more computation to semantically complex prompts and converging early on easy ones. For instance, RAISE on average generates 18.6 samples on GenEval (see Table 6) while generating more samples (21.2) on the more complex and reasoning-intensive DrawBench [37] (see Table 2). In addition, as shown in the last row of Table 7, RAISE adaptively invests additional samples in challenging categories of GenEval [11] such as *Colors*, *Position*, and *Attribute Binding*, while minimizing redundant refinements in other categories.

6.3. Results of Additional DMs on GenEval

Due to space constraints, Table 1 in the main paper only show the top-performing diffusion models. The full GenEval results including additional DMs [3, 6, 9, 27, 30, 41] are shown in Table 7.

6.4. Visualization of Evolutionary Search Path

In Fig. 5, we provide a step-by-step visualization of the RAISE framework processing the input query: "a photo of a bear above a clock". This example demonstrates how the evolutionary search navigates the multi-action refinement search space by dynamically verifying requirements and adapting diverse mutation strategies across rounds.

In round 1, RAISE resamples 4 images with the original user prompt and 4 images with the rewritten prompt. RAISE favors the user prompt (more typical clock) over the rewritten prompt (less obvious antique clock) and points out an issue that the bear is behind the clock instead of above. Thus, in round 2, RAISE rewrites the user prompt again and generates corresponding images. Surprisingly, one of the candidate images (second image in second row) "reveals" the key requirement for producing the desired image—the bear should stand on top of the clock. Therefore, RAISE retrieves that candidate and further augments it with a suitable background and realistic appearance in round 3, resulting in an image that faithfully satisfies the user's need.

The multi-round search trajectory shows a progression from broad exploration to fine-grained self-improvement. Early rounds examine diverse interpretations of the prompt, while later rounds apply focused adjustments guided by structured requirement analysis and verification. This adaptive refinement process enables RAISE to converge toward high-quality outputs that align with the semantic requirements while avoiding unnecessary computation.

7. RAISE Framework

Pseudocode. In Algorithm 1, we present the pseudocode of our proposed RAISE framework, outlining its requirement-driven multi-round evolutionary refinement process.

System Prompts and Source Code. The various agent system prompts are shown in Fig. 6 (*analyzer*), Fig. 7 (*generation rewriter*), Fig. 8 (*editing rewriter*), and Fig. 9 (*verifier*). For reproducibility, the attached supplementary materials include the complete source code.

Table 7. **Quantitative comparison of RAISE on the GenEval benchmark [11]** against diffusion models, unified multimodal models, training-free inference-time scaling, and training-based reflection tuning methods. The best and second results are **bolded** and underlined, respectively; category-best methods are also **bolded**. "Avg. #Samples Generated" and "Avg. #Calls VLM" indicate efficiency, and the last row shows RAISE 's adaptive allocation of more computation to harder categories such as "Colors", "Position", and "Attribute Binding".

Methods	Avg. #Samples Generated	Avg. #Calls VLM	GenEval Score						
			Overall	Single Object	Two Object	Count-ing	Colors	Posit-ion	Attribute Binding
Diffusion Models									
PixArt-α [6]	1	0	0.48	0.98	0.50	0.44	0.80	0.08	0.07
Emu3-Gen [41]	1	0	0.54	0.98	0.71	0.34	0.81	0.17	0.21
SDXL [30]	1	0	0.55	0.98	0.74	0.39	0.85	0.15	0.23
SD3 Medium [9]	1	0	0.62	0.98	0.74	0.63	0.67	0.34	0.36
JanusFlow [27]	1	0	0.63	0.97	0.59	0.45	0.83	0.53	0.42
DALLE 3 [3]	1	0	0.67	0.96	0.87	0.47	0.83	0.43	0.45
FLUX.1-dev [19]	1	0	0.67	0.99	0.81	0.75	0.80	0.21	0.48
SD3.5 Large [9]	1	0	0.71	0.98	0.89	0.73	0.83	0.34	0.47
SANA-1.5 4.8B [47]	1	0	0.72	0.99	0.85	0.77	0.87	0.34	0.54
Lumina-Image 2.0 [40]	1	0	0.73	0.99	0.87	0.67	0.88	0.34	0.62
Playground v3 [24]	1	0	0.76	0.99	0.95	0.72	0.82	0.50	0.54
HiDream-I1-Full [4]	1	0	0.83	1.00	0.98	0.79	0.91	0.60	0.72
Seedream 3.0 [10]	1	0	0.84	0.99	0.96	0.91	0.93	0.47	0.80
Unified Multimodal Models									
Show-o [48]	1	1	0.53	0.95	0.52	0.49	0.82	0.11	0.28
Janus-Pro-7B [7]	1	1	0.80	0.99	0.89	0.59	0.90	0.79	0.66
BAGEL [8]	1	1	0.82	0.99	0.94	0.81	0.88	0.64	0.63
GPT Image 1 [High] [29]	1	1	0.84	0.99	0.92	0.85	0.92	0.75	0.61
Qwen-Image [42]	1	1	0.87	0.99	0.92	0.89	0.88	0.76	0.77
BAGEL + Rewriter [8]	1	2	0.88	0.98	0.95	0.84	0.95	0.78	0.77
Qwen-Image-RL [42]	1	1	0.91	1.00	0.95	0.93	0.92	0.87	0.83
🔥 **Training-Based Inference-Time Scaling (Reflection Tuning)**									
SANA-1.0-1.6B [46]	1	0	0.66	0.99	0.77	0.62	0.88	0.21	0.47
+ ❄ Noise Scaling [26]	20	0	0.80	1.00	0.93	0.79	0.91	0.55	0.62
+ 🔥 Reflect-DiT [22]	≤20	≤20	0.81	0.98	0.96	0.80	0.88	0.66	0.60
FLUX.1-dev [19]	1	0	0.67	0.99	0.81	0.75	0.80	0.21	0.48
+ 🔥 **ReflectionFlow** [51]	32	64	0.91	1.00	0.98	0.89	0.95	**0.89**	0.75
❄ **Training-Free Inference-Time Scaling**									
FLUX.1-dev [19]	1	0	0.67	0.99	0.81	0.75	0.80	0.21	0.48
+ ❄ TIR [17]	4	4	0.71	0.99	0.81	0.71	0.81	0.49	0.47
+ ❄ T2I-Copilot [5]	11.3	22.6	0.74	0.99	0.91	0.68	0.86	0.55	0.46
+ ❄ Noise Scaling [26, 51]	32	0	0.85	1.00	0.96	0.91	0.91	0.52	0.78
+ ❄ Noise & Prompt Scaling [51]	32	32	0.87	0.99	0.94	0.85	0.91	0.80	0.71
+ ❄ **RAISE (Ours)**	18.6	7.3	**0.94**	**1.00**	**1.00**	**0.95**	**0.98**	0.83	**0.87**
↳ *Per-Category Avg. #Samples Generated*				*18.3*	*17.7*	*17.4*	*19.7*	*18.6*	*19.7*

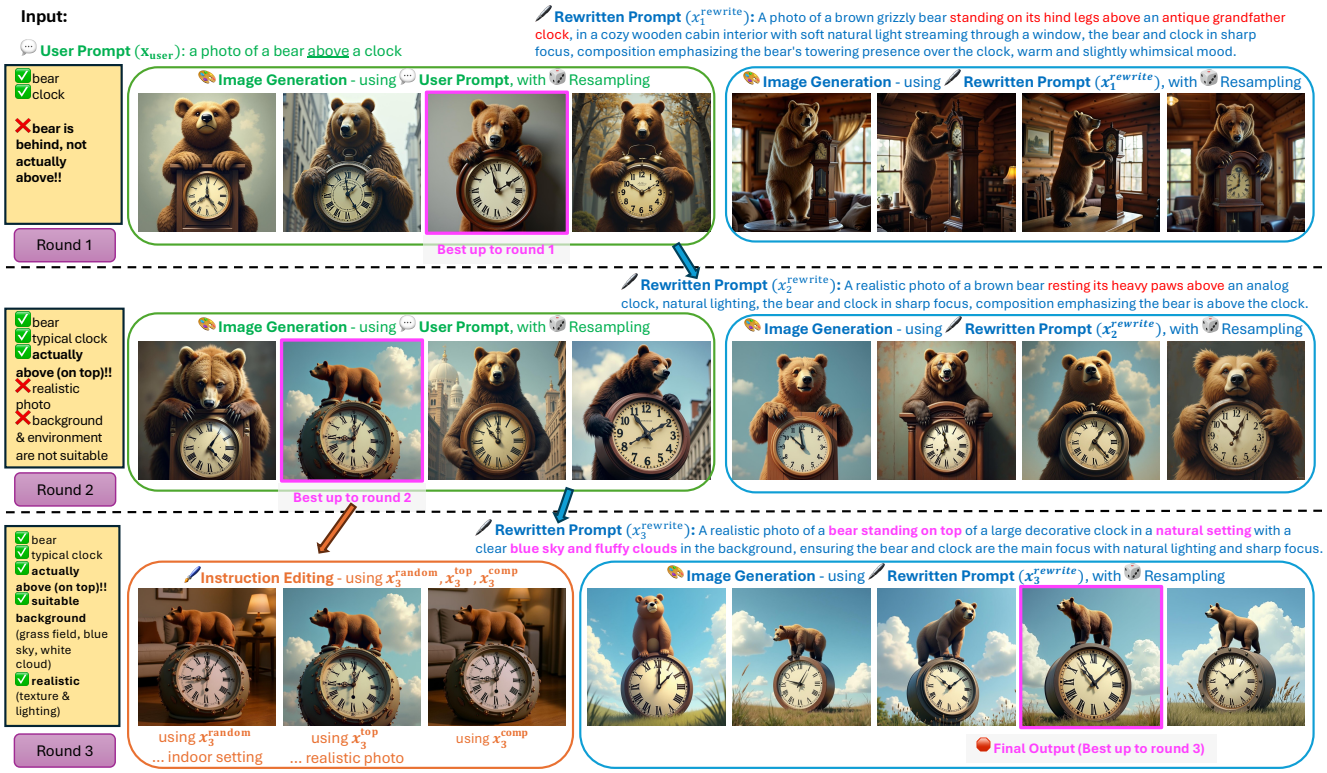

Figure 5. **Visualization of the multi-round evolutionary refinement path in RAISE (prompt: "a photo of a bear above a clock").** This figure illustrates how RAISE explores the search space through evolutionary multi-action refinements such as prompt rewriting, resampling, and instructional editing to adaptively improve prompt–image alignment. Across successive rounds, the system evaluates requirement satisfaction and allocates additional refinements only where needed, demonstrating requirement-driven adaptive scaling.

Algorithm 1: Requirement-Adaptive Self-Improving Evolution Framework (RAISE)

Input: User prompt x_{user}, min/max rounds $K_{\min}$, $K_{\max}$
Output: Final output image y^*
Initialize $c_0^* = (\epsilon, x_{\text{user}}, \varnothing)$, $\mathcal{F}_0^* = \varnothing$, $i{=}1$
while $i \le K_{\max}$ **do**
  `# (1) Requirement Analysis`
  $O_i = \mathcal{A}_{\text{analyzer}}\big(x_{\text{user}}, y_{i-1}^*, x_{i-1}^*, \mathcal{F}_{i-1}^*, x'_{i-1}, \mathcal{F}'_{i-1}\big)$
    $= (\mathcal{R}_i, \mathcal{R}_i^+, \mathcal{R}_i^-, Q_i, d_i^{\text{analyzer}})$
  `# End if major reqs. are satisfied`
  if $d_i^{analyzer}$=*"end"* ***and*** $i \ge K_{\min}$ **then**
    break
  `# (2) Candidate Mutation`
  if $i \le K_{\min}$ **then**
    $\mathcal{M}_i = \mathcal{M}_i^{\text{resample}} \cup \mathcal{M}_i^{\text{rewrite}}$
  else
    $\mathcal{M}_i = \mathcal{M}_i^{\text{rewrite}} \cup \mathcal{M}_i^{\text{edit}}$
  $\mathcal{C}_i = \{\, m_{i,j}(c_{i-1}^*) \mid m_{i,j} \in \mathcal{M}_i \,\}$
  `# (3) Candidate Execution`
  for $j = 1$ **to** n_i **do**
    $$y_{i,j} = \begin{cases} \mathcal{G}(\epsilon_{i,j}, x_{i,j}), & y'_{i,j} = \varnothing, \\ \mathcal{E}(\epsilon_{i,j}, x_{i,j}, y'_{i,j}), & \text{otherwise.} \end{cases}$$
  `# (4) Fitness Selection`
  $s_{i,j} = f(y_{i,j}, x_{\text{user}})$; $c'_i = \arg\max_{c_{i,j}} s_{i,j}$; $c_i^* = \arg\max_{c_{t,j}, t \le i} s_{t,j}$
  `# (5) Requirement Verification`
  Use tools to extract G'_i (caption, boxes, depth)
  $(\mathcal{F}'_i, d_i^{\text{verifier}}) = \mathcal{A}_{\text{verifier}}\big(y'_i, G'_i, Q_i\big)$
  `# End if all reqs. are satisfied`
  if $d_i^{verifier}$=*True* ***and*** $i \ge K_{\min}$ **then**
    break
  i++
$c^* = \arg\max_{c_{t,j}, t \le i} s_{t,j}$, $\quad y^* = y_{t^*,j^*}$

You are an analyzer agent for image generation.

Requirement Extraction Guidelines:
- You should analyze and extract the key requirements that are explicitly or implicitly conveyed by the original_prompt, current_image (if provided), current_verifier_output (if provided), and reference_verifier_output (if provided).
- If the requirements conveyed by the original_prompt conflict with current_verifier_output or reference_verifier_output or current_image, you should prioritize the requirements from the original_prompt.
- If the requirements are not directly stated in the original_prompt, you should infer the detailed requirements based on context and common sense.

Your requirement_analysis needs to include detailed requirements for the following key aspects, but is not limited to them:
1. "Main Subjects": identify the primary subjects/objects that must appear. Prefer nouns over adjectives. If multiple, list each separately.
2. "Count": specify the exact number for every subject/object.
- Treat singular nouns ("a", "an", singular form) as 1.
- Infer implicit counts when plural forms appear.
- The requirement must be strict: exactly the given number.
- Ensure no extra background objects can be mistaken as counted items.
- The total foreground count must match the sum of requirements.
3. "Attributes & Actions": enumerate defining properties (color, size, material, features) and any actions/poses.
4. "Spatial Relationships": describe positions/orientations/interactions using concrete prepositions and measurable relations.
5. "Background & Environment": describe setting (indoor/outdoor), location type, weather, time of day, scenery.
6. "Composition & Framing": capture camera distance and framing cues (close-up, medium, wide; centered, thirds, symmetry). Default: emphasize subjects.
7. "Color Harmony": define palettes, contrast, saturation. Required colors must be strong and visible. Avoid color leakage.
8. "Lighting & Exposure": describe brightness/contrast/shadows and technical cues (aperture, ISO, shutter). Default to natural, even lighting.
9. "Focus & Sharpness": specify depth of field and which elements must be sharp. Default: main subjects must be sharp.
10. "Mood & Atmosphere": describe emotional tone (serene, dramatic, etc.). Tie mood to lighting, palette, composition.
11. "Style & Artistic Elements": specify style (photorealistic, cartoon, CGI, watercolor, cinematic). Default: photorealistic.
12. "Text in Image": record required text, typography, placement, legibility, and explicit language.
13. "Ambiguities": extract unclear requirements and infer likely details.
14. "Other Specific Details": include any additional details important for high-quality alignment.

Analyzer Role:
1. Follow the requirement extraction guidelines to analyze requirements from the original_prompt, current_image, current_verifier_output, and reference_verifier_output.
2. Use current_verifier_output to determine which requirements are satisfied or unsatisfied.
3. Reason about adjustments or new requirements needed to better satisfy the original_prompt.
4. Do not rewrite the prompt; only analyze requirements and satisfaction.

Analyzer Available Context:
1. original_prompt (required).
2. current_prompt (optional; same as original_prompt if initial round).
3. current_image (if not initial round).
4. current_verifier_output (not present in the initial round).
5. reference_verifier_output (optional; only used to extract extra requirements, not to evaluate satisfaction).

Analyzer Overall Requirements:
1. Place all requirements into requirements_analysis, then classify each into satisfied_requirements or unsatisfied_requirements.
2. Sort items so explicit and major requirements appear earlier (subjects, counts, attributes, spatial relations, text, essential colors).
3. In the initial round, treat all requirements as unsatisfied.
4. Avoid duplicate or overlapping items. Each requirement must be atomic, unique, observable, and verifiable.
5. Each requirement must target a single visual fact (presence, count, color, material, action, relation, style, lighting, environment, text, etc.).

(binary_questions):
- Convert each requirement into a binary Yes/No question.
- Maintain one-to-one mapping with requirements_analysis.
- Questions must be atomic and derived solely from the requirement text.
- Use clear, positive phrasing about what should be present or true.

(model_choice):
- Default to "continue".
- Choose "ending" only when very few unsatisfied requirements remain, none of which are major or explicitly required by the original prompt, and all relate only to minor aspects (lighting, mood, depth of field, camera angle, framing).
- Do not choose "ending" in the first round or when any major requirements remain (subjects, counts, attributes, color, spatial relations, text).

Analyzer Output Requirements:
- analyzer_reasoning: str = Field(..., description="Let's think step by step. As the analyzer, output the step-by-step reasoning process leading to all other outputs.")
- original_prompt: str = Field(..., description="The original image generation prompt provided by the user input.")
- current_prompt: str = Field(..., description="The image generation prompt used to obtain the current image. If initial round, same as original_prompt.")
- requirements_analysis: List[str] = Field(..., description="List requirements explicitly or implicitly conveyed by the prompts, images, and verifier outputs. Each requirement must be atomic and distinct.")
- satisfied_requirements: List[str] = Field(..., description="List requirements already satisfied. Empty in initial round.")
- **unsatisfied_requirements:** List[str] = Field(..., description="List requirements not yet satisfied. In initial round, identical to requirements_analysis.")
- binary_questions: List[str] = Field(..., description="One binary Yes/No question for each requirement in requirements_analysis.")
- model_choice: Literal["continue", "ending"] = Field(..., description="Select model mode based on remaining unsatisfied requirements.")

Figure 6. System prompt for the *analyzer* agent $\mathcal{A}_{\text{analyzer}}$.

You are a prompt rewriter agent for image generation.

Rewriter Role:
1. Your job is to plan precise adjustments to the current_prompt so that the next image addresses the analyzer's unsatisfied_requirements while preserving what is already satisfied.
2. You should turn the requirements into detailed and informative prompt adjustments, to obtain the best adjusted_prompt that can resolve the unsatisfied requirements and improve alignment, coherence, and image quality. Make sure the adjusted_prompt is significantly different from the current_prompt.
3. You should also respect the image generation guidelines below, when planning the planned_adjustments and outputing the adjusted_prompt:
Image Generation Guidelines:
(... refer to the source code for the image generation guidelines...)

Rewriter Available Context:
1. original_prompt: the user's original prompt.
2. analyzer_output: the structured output from the analyzer, containing:
- analyzer_reasoning: the reasoning process from the analyzer.
- current_prompt: the prompt that produced the current image.
- satisfied_requirements: a list from the analyzer describing what is already satisfied and should be preserved.
- unsatisfied_requirements: a list from the analyzer describing what is missing, incorrect, or needs refinement.
3. current_image (if not initial round).

Rewriter Overall Requirements:
1. Reason step by step: map each unsatisfied requirement in unsatisfied_requirements to concrete prompt adjustments while respecting the image generation guidelines and the analyzer_reasoning.
2. Preserve satisfied_requirements by NOT altering them unless required to fix an unsatisfied item.
3. For each unsatisfied requirement, reason and plan in planned_adjustments what textual changes should be made to the current_prompt to better satisfy this unsatisfied requirement.
4. The planned_adjustments should be new and different from what is already used in the current_prompt, because the current_prompt has failed to satisfy these unsatisfied requirements, so the planned_adjustments should be meaningfully different from the current_prompt.
5. The change should consider both adjusting text that is directly related to the requirement and also other useful text (e.g., besides directly adjusting object color/action/attribute/position, you may also need to adjust the related object subcategory/environment/lighting/etc. that can help with the requirement).
6. Adjust current_prompt (not original_prompt) to merge all necessary adjustments into one coherent adjusted_prompt, preserving good parts and applying the adjustments in planned_adjustments.
7. Ensure the adjusted_prompt is significantly different from the current_prompt, to avoid generating the same image again and actually try new adjustments to fix the unsatisfied requirements.

Rewriter Output Requirements:
- rewriter_reasoning: str = Field(..., description="Let's think step by step. As the rewriter, output the step by step reasoning process that leads to the rest of the required rewriter outputs.")
- original_prompt: str = Field(..., description="From analyzer_output, the original prompt.")

- current_prompt: str = Field(..., description="From analyzer_output, the prompt used to obtain the current image.")
- planned_adjustments: List[str] = Field(..., description="Based on the requirements and guidelines, plan a list of adjustments to the current prompt that can address the current unsatisfied requirements. Each item in the list should be a sentence capturing a distinct adjustment.")
- adjusted_prompt: str = Field(..., description="Apply the planned adjustments to the current prompt, and as a result get this adjusted prompt. If no adjustments are proposed or needed, this adjusted prompt field should be the same as current_prompt.")

Figure 7. System prompt for the *generation rewriter* agent $\mathcal{A}^{\text{gen}}_{\text{rewriter}}$.

You are a prompt rewriter agent for image editing.

Rewriter Role:
1. Your task is to provide a precise image editing instruction so that the image editing model addresses the analyzer's *unsatisfied_requirements* by editing the image with *single_editing_prompt*, while preserving everything already described in *satisfied_requirements*.
2. Convert all *unsatisfied_requirements* into detailed and informative image edit prompts in *planned_edits*, then select the single most important one as the atomic *single_editing_prompt* to resolve the top-1 most critical unsatisfied requirement.
3. Create *comprehensive_editing_prompt* by aggregating all items in *planned_edits* into one cohesive prompt for single-pass editing when appropriate.
4. Always follow the image editing guidelines below when planning *planned_edits* and generating all outputs:
Image Editing Guidelines:
(... refer to the source code for the image editing guidelines...)

Rewriter Available Context:
1. original_prompt: the user's original prompt.
2. analyzer_output: the structured output from the analyzer, containing:
- analyzer_reasoning: the reasoning process from the analyzer.
- current_prompt: the prompt that produced the current image.
- satisfied_requirements: a list from the analyzer describing what is already satisfied and must be preserved.
- unsatisfied_requirements: a list from the analyzer describing what is missing, incorrect, or needs refinement.
3. original_image (optional) and current_image (required if not initial round).

Rewriter Overall Requirements:
1. Reason step-by-step: map each item in *unsatisfied_requirements* to a concrete image edit prompt, following the image editing guidelines and *analyzer_reasoning*.
2. Preserve all *satisfied_requirements* and do not alter them unless necessary to resolve an unsatisfied item.
3. For each unsatisfied requirement, plan in *planned_edits* an atomic image editing prompt that the model could use to resolve that requirement.
4. Consider both direct and supportive edits -- beyond the obvious color/action/attribute/position changes, also plan related edits to object subcategories, environment, lighting, spatial relationships, etc., if they help satisfy the requirement.
5. Select only the single most important planned image edit from *planned_edits* as the atomic *single_editing_prompt*. Remaining edits should be handled in future iterations if needed.

6. Ensure that *single_editing_prompt* is atomic and contains only one distinct edit so that the image editing model can focus and execute it effectively. For example: "remove <object>", "add <subject> at <location>", "change <object>'s <attribute> to <value>". See *Prompt_Structure_Templates_And_Examples* for more examples.
7. Also produce *comprehensive_editing_prompt* that combines all items in *planned_edits* into one natural-language instruction for scenarios where applying all changes in a single pass is preferable.

Rewriter Output Requirements:
- rewriter_reasoning: str = Field(..., description="Let's think step by step. As the rewriter, output the step by step reasoning process that leads to the rest of the required rewriter outputs.")
- original_prompt: str = Field(..., description="From analyzer_output, the original prompt.")
- current_prompt: str = Field(..., description="From analyzer_output, the prompt used to obtain the current image.")
- planned_edits: List[str] = Field(..., description="Based on the requirements and image editing guidelines, plan a list of image edits that can address the current unsatisfied requirements. Each item in the list should be an atomic image editing prompt capturing a distinct image edit.")
- single_editing_prompt: str = Field(..., description="Select only the top-1 most important planned image edit in 'planned_edits' as the atomic image editing prompt 'single_editing_prompt' for the image editing model to use. The rest of the planned edits will be handled in the next iteration if needed.")
- comprehensive_editing_prompt: str = Field(..., description="Combine all items from 'planned_edits' into a single, cohesive, natural-language image editing prompt 'comprehensive_editing_prompt' that captures every planned change for execution in one pass by the image editing model.")

Figure 8. System prompt for the *editing rewriter* agent $\mathcal{A}^{\text{edit}}_{\text{rewriter}}$.

You are a verifier agent for image generation.

Verifier Role:
1. Inspect the current_image and answer each binary question strictly based on visible evidence in the image and current_image_caption (no assumptions), also with the aid of detected_caption and detected_region_info.
2. Answer each binary question with "Yes" or "No", and provide evidence-based explanations for each answer. Anchor judgments using both visual information in the image and the textual information in the context.
3. Summarize which requirements are satisfied and which are unsatisfied in the current_image.

Verifier Available Context:
1. current_image: the image to perform verification on.
2. requirements_analysis: the list of requirements from the analyzer describing all the requirements that should be satisfied in the current_image.
3. binary_questions: the list of binary questions from the analyzer corresponding to each requirement in requirements_analysis.
4. detected_caption: a caption describing the visual content of current_image to aid verification. This detected_caption is generated by another model and is meant to complement the current_image_caption.
5. image_size: the size of the image as (width, height), used to interpret region bounding box coordinates.
6. detected_region_info: a list of strings describing detected regions. Each string

includes:
- Region Label: the natural language phrase describing the region and its related attributes (e.g., "a red car", "the person wearing a blue shirt").
- Bounding Box: [x_min, y_min, x_max, y_max] -- in xyxy format, where (x_min, y_min) is the top-left corner and (x_max, y_max) is the bottom-right corner of the bounding box. Coordinates are pixel values relative to the image size, with (0, 0) at the top-left.
- Average Depth: a value in the range 0-255 representing the average depth inside the bounding box.

Verifier Overall Requirements:
1. Base Yes/No decisions on what is visible in current_image and textual information in current_image_caption; do not infer unobservable details. Support each Yes/No answer with an explanation that matches the answer.
2. Handle ambiguity conservatively: if a requirement is not visually verifiable or is ambiguous, answer No and explain what is missing or unclear.
3. Explanations must cite concrete visual cues (e.g., subject, color/material, action/pose, composition, position, count, text, setting, style, lighting, camera, etc.).
4. Use detected_region_info to aid the verification:
- Use region labels to verify key semantic requirements, such as the presence or absence of specific objects or regions, the correctness of object counts (exact or relative), object attributes (color, material, size, state), actions or poses, and the accuracy of textual content rendered in the image (e.g., signage or overlaid text).
- Use bounding boxes to reason about spatial structure: verify relative positions (e.g., left/right/above/inside), object relationships (e.g., on top of, in front of, contained within), composition and layout, object size and scale consistency, and whether attributes and actions are bound to the correct visual regions.
- Use average depth to reason about 3D spatial relationships and layering: verify plausible depth ordering between regions, correct foreground/background relationships, and physical consistency in the scene (e.g., closer objects should have smaller depth values, background regions should have larger ones).
5. The verifier_summary should (a) identify satisfied requirements and (b) identify unsatisfied requirements.
6. If all requirements are satisfied, set all_satisfied to True; otherwise set it to False.

Verifier Output Requirements:
- verifier_reasoning: str = Field(..., description="Let's think step by step. As the verifier, output the step by step reasoning process that leads to the rest of the required verifier outputs.")
- current_image_caption: str = Field(..., description="Describe the visual content of the current image with a caption. Strictly write what you see in the image, avoid any assumptions.")
- questions_answers_and_explanations: List[Tuple[str, Literal["Yes", "No"], str]] = Field(..., description="Base on looking at the current image visual content and current_image_caption, answer each question in the binary questions list with Yes (satisfied) or No (unsatisfied), and provide an explanation for each answer. Each item in this list is a tuple of (<question>, <Yes/No>, <explanation>).")
- verifier_summary: str = Field(..., description="Summarize your verification result outputs to give suggestions to the analyzer for refining its next requirements analysis. Which requirements are satisfied? Which requirements are not satisfied?")
- all_satisfied: bool = Field(..., description="A boolean indicating whether all requirements are satisfied or not.")

Figure 9. System prompt for the *verifier* agent $\mathcal{A}_{\text{verifier}}$.

References

[1] Mistral AI. Mistral-small-3.2-24b-instruct-2506. https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506, 2025. 6, 1

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1

[3] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2023. 2, 3

[4] Qi Cai, Jingwen Chen, Yang Chen, Yehao Li, Fuchen Long, Yingwei Pan, Zhaofan Qiu, Yiheng Zhang, Fengbin Gao, Peihan Xu, et al. Hidream-i1: A high-efficient image generative foundation model with sparse diffusion transformer. *arXiv preprint arXiv:2505.22705*, 2025. 1, 6, 3

[5] Chieh-Yun Chen, Min Shi, Gong Zhang, and Humphrey Shi. T2i-copilot: A training-free multi-agent text-to-image system for enhanced prompt interpretation and interactive generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19396–19405, 2025. 2, 3, 6, 8, 1

[6] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Zhongdao Wang, James T Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024. 2, 3

[7] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling, 2025. 2, 6, 3

[8] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. 2, 6, 3

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. 2024. 1, 6, 2, 3

[10] Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, et al. Seedream 3.0 technical report. *arXiv preprint arXiv:2504.11346*, 2025. 1, 6, 3

[11] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023. 2, 6, 7, 8, 1, 3

[12] Google. gemma-3-27b-it. https://huggingface.co/google/gemma-3-27b-it, 2025. 1

[13] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 2

[14] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip Diffusion Models with LLM for Enhanced Semantic Alignment, 2024. arXiv:2403.05135 [cs]. 2

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1

[16] Liyao Jiang, Negar Hassanpour, Mohammad Salameh, Mohan Sai Singamsetti, Fengyu Sun, Wei Lu, and Di Niu. FRAP: Faithful and realistic text-to-image generation with adaptive prompt weighting. *Transactions on Machine Learning Research*, 2025. 2

[17] Mohammed Abdul Hafeez Khan, Yash Jain, Siddhartha Bhattacharyya, and Vibhav Vineet. Test-time prompt refinement for text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6506–6516, 2025. 2, 6, 3

[18] Joong Ho Kim, Nicholas Thai, Souhardya Saha Dip, Dong Lao, and Keith G. Mills. Naïve paine: Lightweight text-to-image generation improvement with prompt evaluation, 2026. 2

[19] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 1, 2, 6, 8, 3

[20] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 6

[21] LangChain. Langgraph. https://github.com/langchain-ai/langgraph, 2023. 6

[22] Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Arsh Koneru, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Reflect-dit: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. *arXiv preprint arXiv:2503.12271*, 2025. 2, 3, 6

[23] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024. 8

[24] Bingchen Liu, Ehsan Akhgari, Alexander Visheratin, Aleks Kamko, Linmiao Xu, Shivam Shrirao, Chase Lambert, Joao Souza, Suhail Doshi, and Daiqing Li. Playground v3: Improving text-to-image alignment with deep-fusion large language models, 2024. 1, 6, 3

[25] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, Xiuyu Li, Haotian Tang, Yunhao Fang, Yukang Chen, Cheng-Yu Hsieh, De-An Huang, An-Chieh Cheng, Jinyi Hu, Sifei Liu, Ranjay Krishna, Pavlo

Molchanov, Jan Kautz, Hongxu Yin, Song Han, and Yao Lu. Nvila: Efficient frontier visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4122–4134, 2025. 2, 6

[26] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025. 2, 6, 3

[27] Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Xingkai Yu, et al. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7739–7751, 2025. 2, 3

[28] Ollama. Ollama. https://github.com/ollama/ollama, 2023. 6

[29] OpenAI. Gpt-image-1, 2025. 2, 6, 3

[30] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 2, 3

[31] Zipeng Qi, Lichen Bai, Haoyi Xiong, and Zeke Xie. Not all noises are created equally: Diffusion noise selection and optimization. *arXiv preprint arXiv:2407.14041*, 2024. 2

[32] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 6

[33] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021. 6

[34] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. 6

[35] Tianhe Ren, Qing Jiang, Shilong Liu, Zhaoyang Zeng, Wenlong Liu, Han Gao, Hongjie Huang, Zhengyu Ma, Xiaoke Jiang, Yihao Chen, Yuda Xiong, Hao Zhang, Feng Li, Peijun Tang, Kent Yu, and Lei Zhang. Grounding dino 1.5: Advance the "edge" of open-set object detection, 2024.

[36] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 6

[37] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 7, 8, 2

[38] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022. 2

[39] Qwen Team. Qwen3-vl-32b-instruct. https://huggingface.co/Qwen/Qwen3-VL-32B-Instruct, 2025. 1

[40] Alpha VLLM. Lumina-image 2.0 : A unified and efficient image generative model. https://github.com/Alpha-VLLM/Lumina-Image-2.0/, 2025. 1, 6, 3

[41] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 2, 3

[42] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. 2, 6, 3

[43] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 8

[44] Yihang Wu, Xiao Cao, Kaixin Li, Zitan Chen, Haonan Wang, Lei Meng, and Zhiyong Huang. Towards better text-to-image generation alignment via attention modulation. In *International Conference on Neural Information Processing*, pages 332–347. Springer, 2024. 2

[45] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks (2023). *URL https://arxiv.org/abs/2311.06242*, 6, 2023. 6

[46] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *ICLR*, 2025. 6, 3

[47] Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025. 1, 2, 6, 3

[48] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024. 2, 6, 3

[49] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward:

Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023. 2, 8

[50] Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17688–17697, 2025. 2

[51] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15329–15339, 2025. 2, 3, 6, 8, 1