RUBRICEVO: ESCAPING UNIFORM REWARDS IN RUBRIC-BASED TEXT-TO-IMAGE REINFORCEMENT LEARNING VIA ONLINE RUBRIC EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Rubric-based rewards have recently gained attention in text-to-image RL, as decomposing prompt-following into atomic, verifiable criteria enables comprehensive verification of complex compositional semantics. However, existing rubric-based methods for text-to-image RL construct their rubric sets before training and keep them fixed throughout training. The fixed rubrics are never checked against the images that the current policy generates, so their difficulty can mismatch the ability of the policy. Specifically, many rubrics are *uniform*: they are satisfied or failed by nearly all rollouts. A uniform rubric assigns nearly the same score to every image in a rollout group. Rewards then carry little difference across samples, leaving the policy with almost no guidance on which samples to reinforce. To address this, we propose ***RubricEvo***, an online rubric evolution framework that revises rubrics at each training step based on feedback from the policy's current rollouts, keeping the rubrics discriminative throughout training. The feedback takes two forms. The pass rate of each rubric identifies the uniform ones and tells the direction to adjust their difficulty. The visual differences among rollout images expose distinctions that the existing rubrics do not cover, from which new rubrics are constructed. Since the two forms of feedback are heterogeneous, handling them in a single MLLM call risks letting the rubric text dominate the visual evidence. RubricEvo therefore processes them with a ***two-stream*** strategy, yielding rubrics both faithful to the prompt and grounded in the actual differences among rollouts. The rollouts are then re-scored under the updated rubrics to restore discriminative rewards. Extensive experiments on widely adopted text-to-image benchmarks show that RubricEvo consistently outperforms state-of-the-art scalar-based and rubric-based methods.

1 INTRODUCTION

Reinforcement learning (RL) has become a standard post-training paradigm for aligning text-to-image (T2I) generation models with human intent (Liu et al., 2025; Xue et al., 2025; Wang et al., 2025b). As prompts grow increasingly compositional, imposing many requirements at once, constructing a reward that faithfully reflects how well an image meets each requirement becomes difficult. Rubric-based rewards address this by decomposing a prompt into a set of atomic, verifiable criteria (e.g., *"Is the dog wearing a raincoat?"*), and scoring each image by the number of criteria it satisfies (Feng et al., 2025; Kao et al., 2026). This yields a fine-grained reward signal for optimizing complex instruction following.

However, existing rubric-based text-to-image RL methods construct rubrics *offline*: the rubrics are built before training begins and kept fixed throughout training. During RL training, a rubric provides a stronger learning signal when it separates better rollouts from worse ones within the same group. Rubrics constructed offline often fail to do so: since they are never checked against the images the current policy generates, their difficulty can mismatch the model's ability. Some criteria fall below the model's ability and are satisfied by every rollout: in Figure 1, all generated images pass *"Is there a dog in the image?"*. Other criteria exceed the model's ability and are failed by every rollout: all images fail *"Is every strand of the dog's fur sharply rendered?"*. In both cases, all images in the group receive the same score on that rubric. We call such rubrics, which nearly all rollouts pass or nearly all fail, *uniform*. When most rubrics for a prompt are uniform, all rollouts receive nearly

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

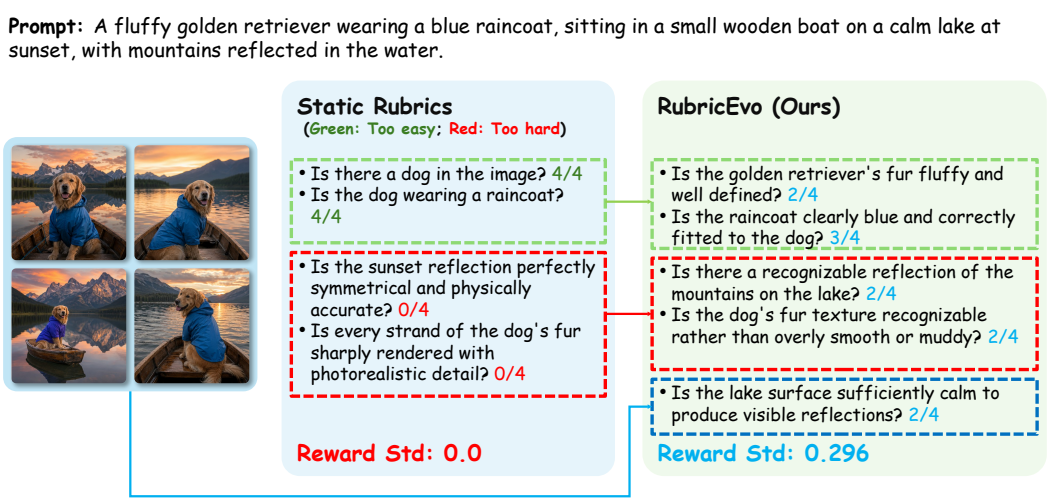

Figure 1: **Example of our proposed rubric evolution.** The training prompt is shown at the top. *Left:* images generated by the current policy. *Middle:* each static rubric is either satisfied by all images or failed by all images. *Right:* RubricEvo refines them online into criteria that separate the images. $x/4$ denotes the number of images (out of 4 rollouts) that satisfy the rubric.

identical total rewards. Since the optimization signal comes from reward differences among samples of the same prompt, low reward variance leaves the policy with almost no guidance on which samples to reinforce (Razin et al., 2025).

Previous visual generation RL methods address low reward variance under scalar reward signals by reducing the contribution of low-variance samples to optimization, either by filtering out such rollout groups (Li et al., 2025b) or lowering the sampling priority of their prompts (Li et al., 2026). However, rubric-based rewards rely on binary criteria and therefore produce uniform group rewards much more frequently; simply excluding these groups would discard a large fraction of rollouts and reduce optimization efficiency. Instead of keeping the rubrics fixed and filtering samples they fail to distinguish, we propose ***RubricEvo***, which actively *evolves* uniform rubrics into criteria tailored to the current rollout group, turning otherwise uninformative rollouts into useful supervision for optimizing the current policy. Specifically, RubricEvo uses a multimodal LLM to exploit two complementary signals: it first uses the original rubric scores to determine whether each criterion is too easy or too hard for the current policy and adjusts its difficulty accordingly, and then directly compares the generated images to discover meaningful visual differences that can be turned into new rubrics. The generator is finally optimized based on the evolved rubrics, which effectively alleviate low reward variance at each training step and improve overall optimization efficiency.

The evolution stage is the core of RubricEvo. When implementing it, we find that directly feeding the uniform rubrics, the prompt, and the rollout images to a single MLLM call, has a clear pitfall. The model largely overlooks the visual differences that exist among the rollout images but are not covered by the existing rubrics. As a result, its outputs are mostly variants of the existing criteria, and few new criteria arise from these uncovered differences. To address this, we adopt a ***two-stream*** design that handles the two inputs separately. A text stream takes only the uniform rubrics and their pass rates, and adjusts their difficulty while staying faithful to the prompt. The image stream, in contrast, receives no original rubrics at all: it first explicitly analyzes the salient differences among the rollout images, and then constructs new rubrics targeting these observed differences. The outputs of the two streams are merged and deduplicated. We find that this design makes full use of the feedback carried by the rollouts, yielding a rubric set that is both faithful to the prompt and grounded in the actual differences among the current generations.

We evaluate RubricEvo on widely adopted T2I benchmarks (Wang et al., 2025a; Hu et al., 2024; Wei et al., 2025). RubricEvo consistently outperforms state-of-the-art scalar reward models, including UnifiedReward (Wang et al., 2025b), UnifiedReward-Flex (Wang et al., 2026), and FIRM (Zhao et al.,

2026), as well as state-of-the-art static rubric-based methods (Kao et al., 2026; Feng et al., 2025), on both in-distribution and out-of-distribution evaluations.

In a nutshell, our contributions are as follows:

- We propose ***RubricEvo***, an online rubric evolution framework that diagnoses uniform rubrics from on-policy rollouts and rewrites them into discriminative criteria, resolving the low reward variance of static rubrics.
- We propose a ***two-stream*** evolution strategy that decouples rubric rewriting from image analysis, yielding rubrics faithful to the prompt and grounded in the differences among current rollouts.
- Extensive experiments on challenging benchmarks show that RubricEvo outperforms state-of-the-art scalar and rubric-based reward methods.

2 Related Works

2.1 Reinforcement Learning for Image Generation

Following the success of RLHF-style (Ouyang et al., 2022) post-training in large language models, reinforcement learning has also been widely adopted in modern text-to-image post-training to better align generated images with human preferences. Early works such as Diffusion-DPO (Wallace et al., 2024) and D3PO (Yang et al., 2024) adapt Direct Preference Optimization (DPO) (Rafailov et al., 2023) to diffusion models. In a different line of research, DPOK (Fan et al., 2023) and DDPO (Black et al., 2024) introduce Proximal Policy Optimization (PPO)-style (Schulman et al., 2017) optimization into image generation. However, these methods are limited to DDPM-based (Ho et al., 2020) diffusion models due to their reliance on stochastic sampling processes. More recently, Flow-GRPO (Liu et al., 2025) and DanceGRPO (Xue et al., 2025) pioneer the replacement of deterministic ODE-based sampling with equivalent SDE formulations, enabling flow-based generative models to be optimized using PPO-style methods. Under this paradigm, subsequent works aim to further improve optimization efficiency by designing more aggressive exploration strategies (Ding & Ye, 2025; He et al., 2025; Li et al., 2025a) or by assigning fine-grained rewards at each timestep (He et al., 2025). While effective, these methods focus on the optimization algorithm and treat the reward signal as a given input.

2.2 Reward Models for Image Generation

Reward models for text-to-image generation have shifted from VLM-based scoring to MLLM-based judging: early VLM-based models such as CLIPScore (Hessel et al., 2021), ImageReward (Xu et al., 2023), HPSv2 (Wu et al., 2023), and PickScore (Kirstain et al., 2023) are trained on human preference data and score an image by global image–text embedding similarity. Among MLLM-based rewards (Zhou et al., 2025; Wu et al., 2025), UnifiedReward (Wang et al., 2025b) unifies pointwise scoring and pairwise ranking in a single model, UnifiedReward-Flex (Wang et al., 2026) further proposes a context-adaptive reasoning trace that instantiates fine-grained criteria for each input, and FIRM (Zhao et al., 2026) trains specialized critics on curated scoring data and modulates the instruction-following score with a quality assessment. Despite these advances, all of them emit a single scalar reward per image: the criteria behind a score remain implicit, and fine-grained compositional errors are easy to overlook. Rubric-based rewards address this by decomposing each prompt into atomic, verifiable criteria and scoring an image by the criteria it satisfies. RubricRL (Feng et al., 2025) directly prompts an LLM to generate a checklist of criteria from each prompt, leaving their quality to the LLM's zero-shot judgment. AutoRubric-T2I (Kao et al., 2026) invests more offline effort, curating a generalized weighted rubric set from human preference pairs before training. In both methods, however, the rubrics are fixed once RL begins: they are never checked against the images the current policy generates, and no amount of offline refinement keeps the reward discriminative within rollout groups.

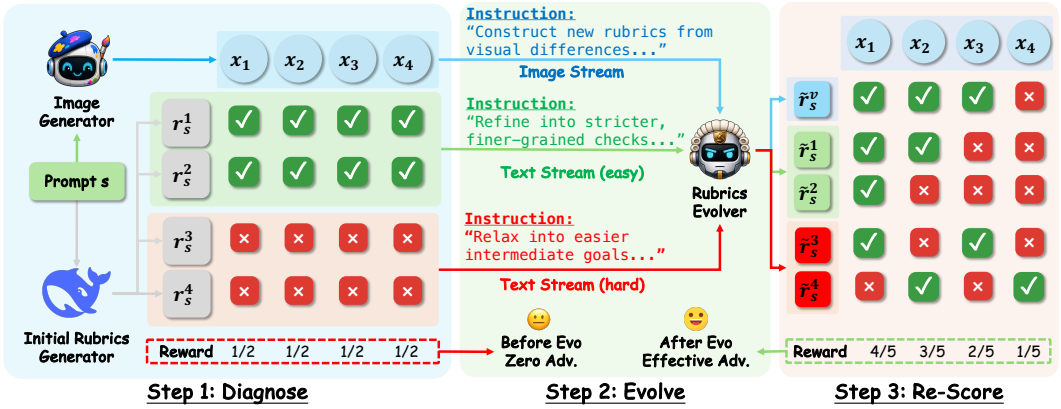

Figure 2: **Overview of RubricEvo.** At each training step, RubricEvo first diagnoses the rubrics r_s^j of prompt s by their pass rates over the current rollouts x_i; here we illustrate an extreme case where all rubrics are uniform so that the group-relative advantage vanishes. The uniform rubrics are then evolved into $\tilde{r}_s^j$ through two decoupled streams: the text stream adjusts their difficulty, while the image stream constructs new rubrics $\tilde{r}_s^v$ from visual differences among rollouts. Finally, all rollouts are re-scored under the evolved rubrics, restoring within-group reward variance.

3 Method

3.1 Problem Formalization

We consider the widely adopted GRPO-based (Guo et al., 2025) reinforcement learning paradigm for text-to-image post-training. Given a prompt s drawn from the training prompt set $\mathcal{D}$, the policy π_θ samples a group of K images:

$$\mathcal{X}(s) = \{x_i\}_{i=1}^{K}, \qquad x_i \sim \pi_\theta(\cdot \mid s). \tag{1}$$

Each image receives a scalar reward $R(x_i, s)$, and the policy is updated according to group-normalized advantages:

$$\hat{A}_i = \frac{R(x_i, s) - \frac{1}{K}\sum_{k=1}^{K} R(x_k, s)}{\mathrm{Std}\left(\{R(x_k, s)\}_{k=1}^{K}\right) + \epsilon}, \tag{2}$$

where ϵ is a small positive constant. Note that $\hat{A}_i$ depends only on how $R(x_i, s)$ differs from the other rewards in the same group. The reward differences among rollouts of the same prompt are therefore the sole source of the training signal. In rubric-based RL methods (Feng et al., 2025; Kao et al., 2026), the reward is decomposed over a prompt-specific rubric set $\mathcal{R}_s = \{r_s^{(j)}\}_{j=1}^{N}$ curated offline:

$$R(x_i, s) = \sum_{j=1}^{N} w_j \cdot \mathbb{I}\left(x_i \models r_s^{(j)}\right), \qquad \sum_{j=1}^{N} w_j = 1, \tag{3}$$

where $x_i \models r_s^{(j)}$ indicates that image x_i satisfies the j-th rubric. Under this decomposition, a rubric contributes to the training signal only through the score differences it creates within the group. If all images in $\mathcal{X}(s)$ receive the same score on $r_s^{(j)}$, this rubric adds a constant to every reward and cancels out in Equation (2). When this holds for most rubrics in $\mathcal{R}_s$, the rewards become nearly identical across the group, and the advantages carry almost no information about which samples to reinforce (Razin et al., 2025).

3.2 Rubrics Evolving

RubricEvo operates at each training step, between rollout sampling and policy update. It proceeds in three stages. *Diagnosis* evaluates every rubric over the current rollout group and identifies the uniform ones. *Evolution* rewrites the uniform rubrics based on feedback from the current rollouts.

Re-score re-evaluates the rollout group with the updated rubric set. Through this process, the reward at every step is computed from rubrics that separate the current samples, keeping the advantage signal in Equation (2) informative throughout training. We detail each stage below.

Diagnosis. Given the rollout group $\mathcal{X}(s) = \{x_i\}_{i=1}^{K}$ and the offline-curated rubric set $\mathcal{R}_s = \{r_s^{(j)}\}_{j=1}^{N}$, we first evaluate each rubric over all images in the group. For each rubric $r_s^{(j)}$, this produces a set of binary rubric scores:

$$\mathbf{y}_j = \{y_{i,j}\}_{i=1}^{K}, \qquad y_{i,j} = \mathbb{I}\Big(x_i \models r_s^{(j)}\Big). \tag{4}$$

We then compute the pass rate of each rubric within the rollout group:

$$p_j = \frac{1}{K}\sum_{i=1}^{K} y_{i,j}. \tag{5}$$

A rubric is considered valid if it is satisfied by some rollouts but failed by others, thus providing non-trivial discrimination among samples. Conversely, a rubric is considered invalid if it becomes uniform, i.e., nearly all rollouts either satisfy or fail it. Formally, we split the initial rubric set as

$$r_s^{(j)} \in \begin{cases} \mathcal{R}_s^{+}, & \alpha < p_j < 1-\alpha, \\ \mathcal{R}_s^{-}, & p_j \le \alpha \ \text{ or } \ p_j \ge 1-\alpha, \end{cases} \tag{6}$$

where α is the uniform detection threshold, $\mathcal{R}_s^{+}$ denotes the valid rubric set, and $\mathcal{R}_s^{-}$ denotes the uniform rubric set. Under this split, every rubric in $\mathcal{R}_s^{+}$ has a within-group score standard deviation above $\sqrt{\alpha(1-\alpha)}$. These rubrics discriminate relatively well among the current rollouts. Rubrics in $\mathcal{R}_s^{-}$ fall below this bound and score the group nearly uniformly. $\mathcal{R}_s^{+}$ is retained for reward computation, while $\mathcal{R}_s^{-}$ is sent to the evolution stage.

Evolution. Given the uniform rubric set $\mathcal{R}_s^{-}$, we employ a multimodal large language model (MLLM) to rewrite it into a new rubric set $\mathcal{R}_s^{\text{new}}$. Following the two-stream design, the two sources of feedback are processed by two independent MLLM calls. The text stream rewrites each uniform rubric r_s^{j} into an evolved counterpart $\tilde{r}_s^{j}$, where the pass rate decides the rewriting mode. Rubrics with $p_j \ge 1-\alpha$ are too easy for the current policy and are refined into stricter, finer-grained checks of the same requirement; rubrics with $p_j \le \alpha$ are too hard and are relaxed into easier intermediate goals. Both modes stay faithful to the prompt s:

$$\mathcal{R}_s^{\text{text}} = \mathcal{E}_{\text{refine}}\left(\mathcal{R}_s^{\text{easy}}, s\right) \cup \mathcal{E}_{\text{relax}}\left(\mathcal{R}_s^{\text{hard}}, s\right), \tag{7}$$

where $\mathcal{R}_s^{\text{easy}}$ and $\mathcal{R}_s^{\text{hard}}$ denote the too-easy and too-hard subsets of $\mathcal{R}_s^{-}$, split by their pass rates as above. The image stream receives no original rubrics. It first analyzes the rollout group $\mathcal{X}(s)$ to identify the salient visual differences among the generated images, and then constructs new rubrics $\tilde{r}_s^{v}$ targeting these observed differences:

$$\mathcal{R}_s^{\text{img}} = \mathcal{E}_{\text{img}}\left(\mathcal{X}(s)\right). \tag{8}$$

In practice, all $\mathcal{E}_*$ reuse the same MLLM that evaluates rubrics for reward computation, and the subscripts only indicate different instructions. Please refer to Appendix for detailed instructions we use. Finally, we merge and deduplicate the outputs of the two streams to obtain the evolved rubric set:

$$\mathcal{R}_s^{\text{new}} = \text{Dedup}\left(\mathcal{R}_s^{\text{text}} \cup \mathcal{R}_s^{\text{img}}\right). \tag{9}$$

The resulting $\mathcal{R}_s^{\text{new}}$ is both faithful to the prompt and grounded in the actual differences among the current rollouts, and is therefore able to separate the images within the rollout group.

Re-score. After obtaining the evolved rubric set $\mathcal{R}_s^{\text{new}}$, we combine it with the retained valid rubrics $\mathcal{R}_s^{+}$ to form a temporary, step-specific rubric set, and re-evaluate the current rollout group $\mathcal{X}(s)$ using the rubric-based reward in Equation (3). The resulting rewards are used to estimate the advantages of images in $\mathcal{X}(s)$ for the current policy update. Importantly, the evolved rubrics are not persisted across training steps. When the same prompt s is sampled again, RubricEvo restarts from its original offline-curated rubric set $\mathcal{R}_s$ and performs diagnosis and evolution again using rollouts from the latest policy. This is because the evolved rubrics are calibrated to the capability and output distribution of the current policy, and may become inappropriate as the policy changes.

Table 1: Evaluation results on UniGenBench++ short. The upper block post-trains SD-3.5-Medium and the lower block post-trains FLUX. Know., Attr., Comp., and Gram. denote World Knowledge, Attribute, Compound, and Grammar. All images are evaluated with the official UniGenBench-EvalModel-72b. UnifiedReward2.0, UnifiedReward-Flex, and FIRM use their 8B variants, while all rubric-based methods use Qwen3-VL-8B. Best results within each base model in bold.

Method	Style	Know.	Attr.	Action	Relation	Comp.	Gram.	Logic	Layout	Text	*Overall*
SD-3.5-Medium (Esser et al., 2024)	90.6	87.0	67.8	59.2	68.8	54.8	60.6	35.3	73.7	14.4	*61.2*
Scalar-Based Rewards											
+ HPSv2 (Wu et al., 2023)	78.6	86.6	67.4	55.0	68.8	56.4	55.5	31.6	73.5	20.7	*59.4*
+ PickScore (Kirstain et al., 2023)	88.9	86.4	67.3	58.8	68.7	53.5	60.7	27.1	73.0	21.3	*60.5*
+ UnifiedReward2.0 (Wang et al., 2025b)	89.2	89.7	69.3	62.8	72.2	60.3	62.0	32.8	79.8	22.1	*64.0*
+ UnifiedReward-Flex (Wang et al., 2026)	87.2	88.3	71.4	72.7	64.6	59.5	62.2	39.5	78.7	20.4	*64.5*
+ FIRM (Zhao et al., 2026)	89.7	89.6	72.0	67.8	73.5	67.5	**64.3**	40.1	82.1	22.4	*66.9*
Rubric-Based Rewards											
+ RubricRL (Feng et al., 2025)	90.7	91.8	74.6	67.7	74.2	66.1	61.2	40.1	82.8	22.7	*67.2*
+ RubricRL+Rejection Sampling	91.4	**91.9**	74.2	69.7	73.5	63.8	64.0	37.6	82.1	25.6	*67.4*
+ RubricRL+CGPO (Li et al., 2026)	**91.5**	90.8	75.0	69.6	72.7	64.2	62.4	38.1	80.4	22.4	*66.7*
+ AutoRubric-T2I (Kao et al., 2026)	90.9	90.7	75.2	67.7	72.5	64.7	63.6	38.1	81.7	**26.2**	*67.1*
+ RubricEvo (Ours)	90.5	91.1	**77.3**	**73.5**	**77.5**	**72.8**	63.1	**41.5**	**83.4**	24.7	***69.6***
FLUX.1-dev (Black Forest Labs, 2024)	85.9	86.9	65.5	62.5	67.3	45.8	**59.2**	27.8	71.5	31.3	*60.35*
Rubric-Based Rewards											
+ RubricRL (Feng et al., 2025)	79.1	89.7	80.6	77.3	76.9	67.1	56.8	45.2	83.2	36.2	*69.21*
+ RubricEvo (Ours)	**90.0**	**91.9**	**82.7**	**79.8**	**81.8**	**75.1**	56.3	**51.6**	**85.3**	**43.1**	*73.77*

4 EXPERIMENTS

4.1 SETTINGS

In this section we discuss the core settings in our experiment, while providing the comprehensive settings in Appendix.

Datasets and Benchmarks. We train on the training split of UniGenBench++ (Wang et al., 2025a), which contains 10,000 compositionally complex text-to-image prompts. We evaluate on UniGenBench++ (Wang et al., 2025a) as the in-domain benchmark, and TIIF-Bench (Wei et al., 2025) as out-of-domain benchmarks.

Base Model and RL Settings. We apply Flow-GRPO (Liu et al., 2025) to Stable Diffusion 3.5 Medium (SD3.5M) (Esser et al., 2024) and FLUX (Black Forest Labs, 2024), following the official hyperparameters. Images are generated at 512×512, with 24 rollouts per prompt. We train with AdamW at a learning rate of $1e{-}4$. For the aggregation weights of each rubric, we follow RubricRL (Feng et al., 2025) to use the uniform weight among rubrics of a prompt, while AutoRubric-T2I (Kao et al., 2026) adopts the offline learned coefficients. For rubrics evolution, we fix uniform rubrics detection threshold $\alpha = 0.1$ unless otherwise specified.

External Model Usage. Before RL training, we use DeepSeek-R1 (Guo et al., 2025) to generate 10 initial rubrics for each training prompt, which are rubrics set for RubricRL (Feng et al., 2025) and initial rubrics set for our method. AutoRubric-T2I (Kao et al., 2026) uses its officially learned rubric set, instead. During training, all rubric-based methods employ Qwen3-VL-8B-Instruct (Bai et al., 2025a) as unified interface for scoring, rubrics evolution and deduplication.

4.2 COMPARISON RESULTS

Comparison with Existing Rewards. We compare RubricEvo with three families of reward models on UniGenBench++ (Wang et al., 2025a): VLM-based scalar rewards (HPSv2 (Wu et al., 2023), PickScore (Kirstain et al., 2023)), MLLM-based scalar rewards (UnifiedReward2.0 (Wang et al., 2025b), UnifiedReward-Flex (Wang et al., 2026), FIRM (Zhao et al., 2026)), and rubric-based rewards (RubricRL (Feng et al., 2025), AutoRubric-T2I (Kao et al., 2026)). Despite using only an off-the-shelf Qwen3-VL-8B-Instruct (Bai et al., 2025a), RubricEvo achieves the best overall score of 69.6 on

Table 2: **Out-of-Distribution Evaluation results on TIIF-Bench**, with the short-prompt and long-prompt splits shown side by side (short on the left, long on the right). All methods use SD-3.5-Medium. Attr, Rel, Rsn, Sty, Dsgn, and Ovr. denote Attribute, Relation, Reasoning, Style, Designer-Level Following, and Overall, and Text denotes Text Generation. A+R, A+Rs, and R+Rs are the corresponding compositional splits. UniRwd abbreviates UnifiedReward. Results are evaluated on Qwen2.5-VL-72B following official setting. Best results within each split in bold.

	TIIF (short)										TIIF (long)									
Method	**Attr**	**Rel**	**Rsn**	**A+R**	**A+Rs**	**R+Rs**	**Text**	**Sty**	**Dsgn**	***Ovr.***	**Attr**	**Rel**	**Rsn**	**A+R**	**A+Rs**	**R+Rs**	**Text**	**Sty**	**Dsgn**	***Ovr.***
SD-3.5-Medium	80.5	79.9	69.5	66.7	63.8	62.4	81.0	73.3	73.9	*72.3*	**83.5**	76.8	68.5	69.7	68.6	66.4	64.7	**76.7**	75.0	*72.2*
Scalar-Based Rewards																				
+ HPSv2	78.0	78.8	72.6	69.0	64.1	65.0	70.1	66.7	72.0	*70.7*	79.0	75.8	75.3	69.3	65.2	65.2	53.9	53.3	72.0	*67.7*
+ PickScore	81.5	77.5	69.5	62.2	59.9	60.3	69.2	**80.0**	73.5	*70.4*	69.0	77.5	69.5	67.1	64.9	67.1	52.5	73.3	74.6	*68.4*
+ UniRwd2.0	82.0	80.6	**75.7**	71.8	63.9	62.4	82.4	73.3	77.2	*74.4*	78.0	75.7	76.8	68.7	63.2	**70.1**	62.4	73.3	77.6	*71.8*
+ UniRwd-Flex	84.0	81.6	68.5	72.5	62.6	65.8	84.6	73.3	78.4	*74.6*	83.0	80.6	73.7	73.1	64.4	63.5	67.9	70.0	78.4	*72.7*
+ FIRM	81.0	79.8	70.0	65.1	64.4	63.7	81.5	70.0	80.6	*72.9*	79.5	77.2	77.2	**73.3**	71.1	66.5	64.7	66.7	81.7	*73.1*
Rubric-Based Rewards																				
+ RubricRL	79.0	79.9	72.0	**75.5**	64.8	65.6	81.0	73.3	76.5	*74.2*	79.0	80.8	77.3	71.7	68.6	68.1	67.9	70.0	76.5	*73.3*
+ RubricRL+Rejection Sampling	79.0	82.7	72.5	70.3	**68.0**	65.3	77.8	**80.0**	81.7	*75.3*	80.0	76.5	75.2	69.8	65.1	67.5	61.5	73.3	81.7	*72.3*
+ RubricRL+CGPO	79.0	81.4	71.0	71.7	64.2	62.2	76.9	73.3	76.9	*73.0*	78.5	77.1	72.2	69.2	66.8	69.5	58.4	70.0	75.0	*70.7*
+ AutoRubric-T2I	**84.5**	82.3	72.0	67.9	59.8	66.6	**88.2**	**80.0**	77.2	*75.4*	**83.5**	**80.9**	76.3	72.4	67.0	67.3	71.5	70.0	76.9	*74.0*
+ RubricEvo (Ours)	84.0	**84.0**	74.6	72.3	65.6	**68.7**	82.8	73.3	**84.3**	***76.6***	82.0	78.0	**78.9**	68.1	**72.8**	66.3	**72.4**	70.0	**82.5**	***74.6***

SD-3.5-Medium, outperforming the strongest scalar baseline FIRM by 2.7 points, RubricRL by 2.4 points, and AutoRubric-T2I by 2.5 points. RubricEvo also obtains the highest overall scores on both the short and long splits of the out-of-distribution TIIF-Bench, suggesting that the improvement is not restricted to the UniGenBench++ training distribution.

Comparison with Low-Variance Avoidance Strategies. We further compare RubricEvo with two strategies that avoid low-variance supervision without modifying the rubrics themselves. RubricRL+Rejection Sampling simply removes uniform rubrics, while CGPO (Li et al., 2026) reduces the sampling probability of prompt groups exhibiting low reward variance. These strategies improve or preserve training efficiency by spending less optimization budget on uninformative signals, but they do not recover supervision from the affected prompt groups: once binary rubric outcomes become saturated, discarding the corresponding rubrics or sampling those groups less frequently cannot create new distinctions among the current rollouts. Consequently, Rejection Sampling and CGPO reach only 67.4 and 66.7 overall, respectively, compared with 69.6 for RubricEvo. In contrast, RubricEvo directly repairs the source of low variance by evolving saturated rubrics according to the current rollouts, turning otherwise uniform binary feedback into discriminative supervision and allowing these prompt groups to continue contributing effective optimization signals.

Improvements over Different Base Models. The lower block of Table 1 repeats the comparison with FLUX (Black Forest Labs, 2024) as the base model. Both blocks use the same initial rubric set and weighting for RubricRL and RubricEvo, so the difference between them reflects only whether the rubrics evolve. On SD-3.5-Medium, RubricEvo improves over the base model and RubricRL by 8.4 and 2.4 points overall, respectively. At optimization step 48,000 on FLUX, RubricEvo scores 73.77, compared with 60.35 for the base model and 69.21 for RubricRL, giving gains of 13.42 and 4.56 points, respectively. Evolving rubrics with the policy thus improves both base models.

4.3 Ablation Study

Effectiveness of Rubric Evolution. We first compare RubricEvo with the *w/o evolution* variant, which keeps the initial rubric set fixed throughout training. As shown in the left of Figure 3, RubricEvo stays above this variant over the entire training process. This shows that rubric evolution consistently improves the effectiveness of RL optimization compared to static rubrics RL.

Effectiveness of the Two-Stream Design. We then ablate the two-stream design with three variants. As shown in the left of Figure 3, removing either stream degrades the performance: *w/o image stream* and *w/o text stream* both fall below the full method, showing that the two sources of feedback are complementary rather than redundant. Moreover, *Coupled*, which processes both sources in a single

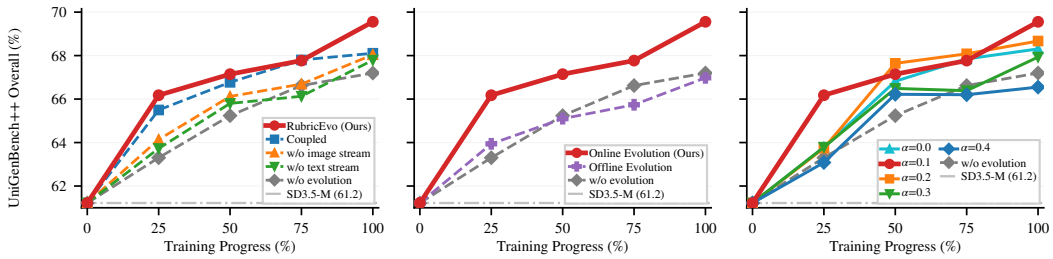

Figure 3: **Detailed analysis of our method.** We report the test set performance on UniGenBench++ over the training process of our method and corresponding variants. **Left**: Ablations on two-stream evolving strategy. **Middle**: Ablation of online evolution strategy. **Right**: Sensitivity analysis of uniform rubrics detection threshold α.

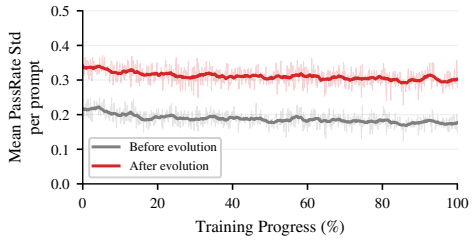

Figure 4: **Per-prompt rubric discriminability during training.** We average rubric PassRate Std within each prompt, then equally across prompts at each training step, before and after evolution.

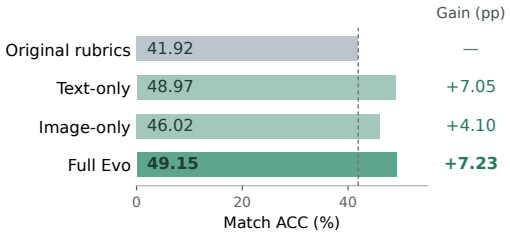

Figure 5: **Preference alignment with Gemini-3.8-Flash before and after rubric evolution.** Match ACC measures how often rubric-set rewards predict the same preference as Gemini-3.8-Flash for an image pair (A better, tie, or B better).

MLLM call, also lags behind RubricEvo, proving that two-stream decoupled evolving is better than multimodal coupled evolution.

Effectiveness of Online Evolution. To verify that rubrics indeed co-evolve with the policy, we compare our *Online Evolution* with *Offline Evolution*, which evolves the rubrics only once using rollouts from the initial policy and then keeps them fixed throughout training. As shown in the middle panel of Figure 3, the two variants' gap consistently widens as the policy is updated. Offline Evolution eventually performs close to the variant without evolution, suggesting that rubrics adapted to the initial policy remain trapped in its local output distribution and gradually become outdated. In contrast, Online Evolution continuously tracks the changing policy, confirming that the rubrics and policy mutually evolve throughout training.

4.4 More Analysis

In this section, we investigate what RubricEvo actually affects by analyzing how the score distribution of rubrics changes before and after evolution. We also investigate the performance-computation cost of RubricEvo.

Rubric Discriminability before and after Evolution. For each prompt, we average the rubric-level PassRate Std $\sqrt{p_j(1-p_j)}$ over its rubric set, where p_j is defined in Equation (5). Figure 4 then averages this statistic equally across prompts at each training step. Evolution increases this per-prompt discriminability across all recorded steps over the training process. We separately examine aggregate reward variance across evolution strategies in Figure 6. Results show that each strategy can improve the reward variance.

Reliability of Evolved Rubric Rewards. We test whether evolved rubrics favor higher-quality images, rather than merely alleviating low reward variance by introducing irrelevant, noisy questions.

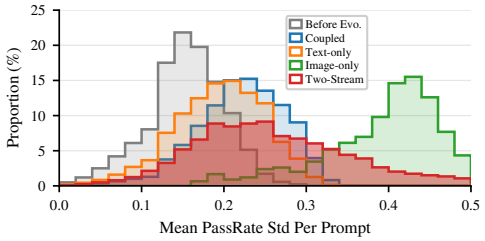

Figure 6: **Distributions of rubric discriminability under different evolution strategies.** For each training prompt we average the PassRate Std over its evolved rubrics, and plot the distribution of this per-prompt statistic across all prompts. Gray is the shared pre-evolution starting point.

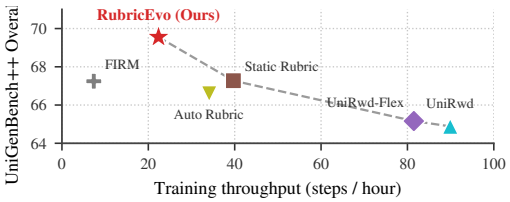

Figure 7: **Performance–throughput trade-off on UniGenBench++.** Each point denotes a reward scheme, plotted by overall score against training throughput. The dashed line connects the methods to illustrate the trade-off between accuracy and throughput.

To this end, we compare their pairwise rankings with blinded Gemini-3.8-Flash judgments on 100 sampled prompt groups. Figure 5 holds the images and reference labels fixed across rubric variants. We measure exact agreement on {A better, tie, B better}, first within each prompt and then equally across prompts. Full Evo increases accuracy from 41.92% to 49.15%, a gain of 7.23 percentage points. This provides evidence that evolution improves the reliability of reward rankings relative to static rubrics under this independent-model audit. Text-only and Image-only reach 48.97% and 46.02%, respectively.

Contribution of Each Feedback Source. Figure 6 examines how the feedback strategies change aggregate reward statistics. Coupled and Text-only occupy similar low-variance regions, while Two-Stream shifts toward higher variance and Image-only has a broader right tail. This is because two-stream methods more balancedly take the text and visual feedback into account, rather than largely relying on text-stream, as the coupled variant does. Figure 3 provide the performance evidence for separating text and visual feedback.

Computational Cost Analysis. Figure 7 plots overall score against training throughput. Single-score rewards such as UnifiedReward are the fastest, but rubric-based rewards evaluate multiple criteria and reach clearly higher accuracy. Within rubric-based methods, online evolution costs some throughput relative to static rubrics and returns a consistent accuracy gain. RubricEvo is also faster than FIRM while scoring higher, since it answers short atomic questions through parallel binary judgments instead of generating a chain-of-thought analysis per criterion. Overall, RubricEvo attains the highest accuracy at a moderate throughput cost.

5 Conclusion

Rubric-based rewards are typically constructed before RL and then held fixed. We showed that as the policy improves, more rubrics end up passed or failed by every rollout in a group, so they stop separating better generations from worse ones and group-relative optimization loses signal. RubricEvo addresses this by evolving rubrics on-policy, identifying the ones that no longer discriminate and rewriting them from the policy's current rollouts. We further found that feeding rubric text and rollout images to a single MLLM call leaves the visual evidence largely unused, so we decouple the two into independent evolution streams. RubricEvo improves SD-3.5-Medium by 8.4 points on UniGenBench++, and the gains transfer to FLUX and to out-of-distribution benchmarks. More broadly, our results suggest that a reward is better maintained during training than fixed before it, and rubrics make this practical because they are explicit enough to be inspected and rewritten as training proceeds.

AI USE STATEMENT

In this work, we used generative AI tools for literature review and writing refinement. All AI-assisted content was reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

RubricEvo improves text-to-image generation and may therefore amplify risks of misleading imagery and harmful or biased content. Rubric generation and evaluation also inherit limitations and biases of the underlying language and vision-language models; improved benchmark scores do not establish safety or fairness. Deployment should include appropriate content safeguards and respect the licenses and usage conditions of the underlying models and data.

REPRODUCIBILITY STATEMENT

Section 3 specifies rubric diagnosis, two-stream evolution, and re-scoring; Section 4.1 identifies the training data, backbones, and external models. Appendices A.1 and A.2 provide training hyperparameters and benchmark evaluation protocols. Appendix A.4 describes baseline implementations, and Appendix A.6 provides the prompt templates used for rubric generation, evolution, and judging.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

Zheng Ding and Weirui Ye. Treegrpo: Tree-advantage grpo for online rl post-training of diffusion models. *arXiv preprint arXiv:2512.08153*, 2025.

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. In *Forty-first international conference on machine learning*, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

Xuelu Feng, Yunsheng Li, Ziyu Wan, Zixuan Gao, Junsong Yuan, Dongdong Chen, and Chunming Qiao. Rubricrl: Simple generalizable rewards for text-to-image generation. *arXiv preprint arXiv:2511.20651*, 2025.

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

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. *arXiv preprint arXiv:2508.04324*, 2025.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment, 2024.

Kuei-Chun Kao, Daixuan Huo, Yuanhao Ban, and Cho-Jui Hsieh. Autorubric-t2i: Robust rule-based reward model for text-to-image alignment. *arXiv*, 2026.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Baoteng Li, Xianghao Zang, Xinran Wang, Xiangyu Na, Zhixiang He, Hao Sun, Chi Zhang, Zhongjiang He, Tianwei Cao, Kongming Liang, and Zhanyu Ma. Curriculum group policy optimization: Adaptive sampling for unleashing the potential of text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 561–571, June 2026.

Chenliang Li, Haiyang Xu, Junfeng Tian, Wei Wang, Ming Yan, Bin Bi, Jiabo Ye, He Chen, Guohai Xu, Zheng Cao, et al. mplug: Effective and efficient vision-language learning by cross-modal skip-connections. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 7241–7259, 2022.

Yuming Li, Yikai Wang, Yuying Zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. Branchgrpo: Stable and efficient grpo with structured branching in diffusion models. *arXiv preprint arXiv:2509.06040*, 2025a.

Zongjian Li, Zheyuan Liu, Qihui Zhang, Bin Lin, Feize Wu, Shenghai Yuan, Zhiyuan Yan, Yang Ye, Wangbo Yu, Yuwei Niu, et al. UniWorld-V2: Reinforce image editing with diffusion negative-aware finetuning and MLLM implicit feedback. *arXiv preprint arXiv:2510.16888*, 2025b.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Yibin Wang, Zhimin Li, Yuhang Zang, Jiazi Bu, Yujie Zhou, Yi Xin, Junjun He, Chunyu Wang, Qinglin Lu, Cheng Jin, et al. Unigenbench++: A unified semantic evaluation benchmark for text-to-image generation. *arXiv preprint arXiv:2510.18701*, 2025a.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation. *arXiv preprint arXiv:2503.05236*, 2025b.

Yibin Wang, Yuhang Zang, Feng Han, Jiazi Bu, Yujie Zhou, Cheng Jin, and Jiaqi Wang. Unified personalized reward model for vision generation, 2026. URL https://arxiv.org/abs/2602.02380.

Xinyu Wei, Jinrui Zhang, Zeqing Wang, Hongyang Wei, Zhen Guo, and Lei Zhang. Tiif-bench: How does your t2i model follow your instructions? *arXiv preprint arXiv:2506.02161*, 2025.

Jie Wu, Yu Gao, Zilyu Ye, Ming Li, Liang Li, Hanzhong Guo, Jie Liu, Zeyue Xue, Xiaoxia Hou, Wei Liu, et al. Rewarddance: Reward scaling in visual generation. *arXiv preprint arXiv:2509.08826*, 2025.

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.

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.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8941–8951, 2024.

Xiangyu Zhao, Peiyuan Zhang, Junming Lin, Tianhao Liang, Yuchen Duan, Shengyuan Ding, Changyao Tian, Yuhang Zang, Junchi Yan, and Xue Yang. Trust your critic: Robust reward modeling and reinforcement learning for faithful image editing and generation, 2026. URL https://arxiv.org/abs/2603.12247.

Shijie Zhou, Ruiyi Zhang, Huaisheng Zhu, Branislav Kveton, Yufan Zhou, Jiuxiang Gu, Jian Chen, and Changyou Chen. Multimodal llms as customized reward models for text-to-image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19638–19648, 2025.

A Appendix

A.1 Training Settings

For a controlled comparison, all reward-model experiments on SD3.5-Medium (Esser et al., 2024) and FLUX.1-dev (Black Forest Labs, 2024) use the backbone-specific optimization settings summarized in Table A1; only the reward signal is changed across methods for a given backbone. Both backbones are optimized with Flow-GRPO (Liu et al., 2025) and parameter-efficient LoRA finetuning (Hu et al., 2021), using the same image resolution, rollout group size, optimizer, learning rate, and number of GRPO inner epochs. Their sampling schedules, KL coefficients, LoRA configurations, and mixed-precision formats follow the corresponding backbone-specific settings.

A.2 Evaluation Settings

We use the same inference configuration for all reward-model variants of a given backbone. Table A1 summarizes the parameters used to generate images for UniGenBench++, DPG-Bench, and TIIF-Bench. We generate four images per prompt with fixed seeds and use the pretrained FlowMatch Euler scheduler associated with each backbone. No stochastic SDE noise is injected during any benchmark inference run.

UniGenBench++. We follow the official UniGenBench++ (Wang et al., 2025a) evaluation pipeline and use its released offline evaluator, UniGenBench-EvalModel-qwen-72b-v1, for all reported results. The evaluator is distilled from Gemini 2.5 Pro and is designed to reproduce the benchmark's fine-grained semantic judgments across its evaluation dimensions. The benchmark authors report an average agreement of over 94% with Gemini 2.5 Pro, making the released evaluator a reproducible substitute for the proprietary judge. We use the official prompts, test points, response parsing, and score aggregation without modification in all comparisons.

DPG-Bench. We use the official DPG-Bench evaluation protocol released with ELLA (Hu et al., 2024). Each prompt is decomposed into dependency-aware natural-language propositions, and generated images are evaluated with the official mPLUG visual-question-answering model (Li et al., 2022) released with the benchmark. We use the official proposition dependencies, category definitions, and aggregation script, and evaluate four generated images per prompt as prescribed by the official benchmark protocol.

TIIF-Bench. We follow the official TIIF-Bench (Wei et al., 2025) evaluation code and use Qwen2.5-VL-72B-Instruct (Bai et al., 2025b) as the evaluation model, as recommended by the official repository. We retain the benchmark-provided evaluation prompts, instruction categories, answer parsing, and score aggregation for both short and long prompt settings.

Model serving. All locally hosted LLM- or MLLM-as-judge models used for reward computation and benchmark evaluation are served with vLLM (Kwon et al., 2023) through an OpenAI-compatible interface. This includes the rubric judge used during RL, the offline UniGenBench++ evaluator, and the Qwen2.5-VL-72B-Instruct evaluator used for TIIF-Bench. Serving changes only inference throughput and does not modify the evaluation prompts, model weights, decoding outputs, or benchmark scoring logic.

A.3 DPG-Bench Results

Table A2 reports out-of-distribution results on DPG-Bench (Hu et al., 2024) using the same SD3.5-Medium policies trained on UniGenBench++. RubricEvo achieves the highest overall score of 87.0 among the compared methods. The evaluation protocol is described in Appendix A.2.

A.4 Baseline Implementations

RubricRL. We reproduce RubricRL (Feng et al., 2025) by using DeepSeek-R1 (Guo et al., 2025) to construct prompt-specific rubrics for every prompt in the UniGenBench++ (Wang et al., 2025a) training set before policy optimization. Specifically, for each text-to-image prompt, DeepSeek-R1

Table A1: Backbone-specific training and benchmark-inference settings used throughout our comparisons.

Setting	SD3.5-Medium	FLUX.1-dev
Training		
Image resolution	512×512	512×512
Rollouts per prompt	24	24
Sampling steps	10	6
Classifier-free guidance scale	4.5	3.5
SDE noise level	0.7	0.9
Batch size per accelerator	3	3
Gradient accumulation steps	4	24
Optimizer	AdamW	AdamW
Learning rate	1×10^{-4}	1×10^{-4}
Adam coefficients	$(0.9, 0.999)$	$(0.9, 0.999)$
Weight decay	10^{-4}	10^{-4}
Gradient clipping	1.0	1.0
GRPO inner epochs	1	1
PPO clipping range	10^{-4}	10^{-4}
KL coefficient	0.02	0.02
EMA	Enabled	Enabled
LoRA rank / scaling	$32/64$	$64/128$
Mixed precision	FP16	BF16
Benchmark Inference		
Image resolution	512×512	512×512
Images per prompt	4	4
Sampling steps	40	30
Classifier-free guidance scale	7.0	3.5
Noise scheduler	FlowMatch Euler	FlowMatch Euler
SDE noise injection	Disabled	Disabled
Random seeds	3407–3410	3407–3410
Inference precision	FP16	BF16

Table A2: **Out-of-Distribution results on DPG-Bench.** All methods use SD-3.5-Medium. Best results in bold.

Method	Attr.	Entity	Global	Other	Relation	*Overall*
SD-3.5-Medium	87.9	90.0	84.8	78.0	92.8	*83.5*
Scalar-Based Rewards						
+ HPSv2	**91.9**	90.7	88.7	88.8	91.1	*86.3*
+ PickScore	88.8	90.9	88.9	89.7	91.4	*85.0*
+ UnifiedReward2.0	90.9	92.1	86.3	**91.6**	90.4	*86.5*
+ UnifiedReward-Flex	90.5	90.9	91.0	**91.6**	91.0	*86.1*
+ FIRM	91.6	92.2	90.7	88.5	89.3	*86.5*
Rubric-Based Rewards						
+ RubricRL	91.7	92.6	87.5	89.2	90.9	*86.5*
+ RubricRL+CGPO	89.8	91.7	85.7	87.2	**94.6**	*86.6*
+ AutoRubric-T2I	90.1	92.1	**91.3**	90.3	91.7	*86.2*
+ RubricEvo (Ours)	91.5	**92.9**	85.6	89.9	92.1	***87.0***

decomposes the requested content into ten atomic, visually verifiable binary questions covering the relevant subjects, attributes, counts, relations, actions, scene, text, and style. The resulting questions are stored with the corresponding prompt and remain fixed throughout RL training. At reward evaluation, each rollout is judged independently against all ten questions using Qwen3-VL-8B-Instruct (Bai et al., 2025a) and Listing 7; their mean binary score defines the RubricRL reward.

AutoRubric-T2I. For AutoRubric-T2I (Kao et al., 2026), we do not rerun its offline rubric-learning pipeline. Instead, we directly use the learned global rubrics and coefficients from its Qwen3-VL-8B, HPSv3-derived released rubric set. This set contains 20 global rules and is applied to every prompt without updates during policy optimization. For each prompt–image pair, we use Listing 8 with Qwen3-VL-8B-Instruct (Bai et al., 2025a), temperature 0.0, and one output token, then read the probability of the `Yes` token rather than the sampled string. The final reward is the weighted sum of these rule-level `Yes`-token probabilities using the released coefficients. In contrast, RubricRL and RubricEvo evaluate ten prompt-specific rubrics per image; this different number of VLM calls also contributes to the throughput differences between the methods.

RubricEvo (Ours). Except for the online evolution stage, RubricEvo uses exactly the same implementation as our RubricRL reproduction. At each policy update, rubrics with pass rates in $[0.1, 0.9]$ are retained, whereas the remaining rubrics are identified as too easy or too hard. The text stream refines too-easy rubrics with Listing 2 and relaxes too-hard rubrics with Listing 3. The visual stream first localizes differences among the current rollouts with Listing 4, then converts them into new atomic binary questions with Listing 5. The visual stream is deliberately blind to the text prompt, preventing it from collapsing into text rewriting; prompt relevance is supplied by the text stream and enforced again when the two streams are merged and deduplicated with Listing 6. The resulting questions are evaluated with the same judge as RubricRL (Listing 7). Because evolution may change the rubric-set size, we recompute the uniform aggregation weights according to the updated number of rubrics. All other reward evaluation and policy optimization procedures remain unchanged from RubricRL in our experiments.

A.5 Qualitative Results

We further provide diverse qualitative results in Figures A2 and A3.



Figure A1: **Qualitative results of RubricEvo.** Representative generations from FLUX.1-dev (blue frames) and SD3.5-Medium (red frames). Images are randomly sampled from the final evaluated checkpoints.

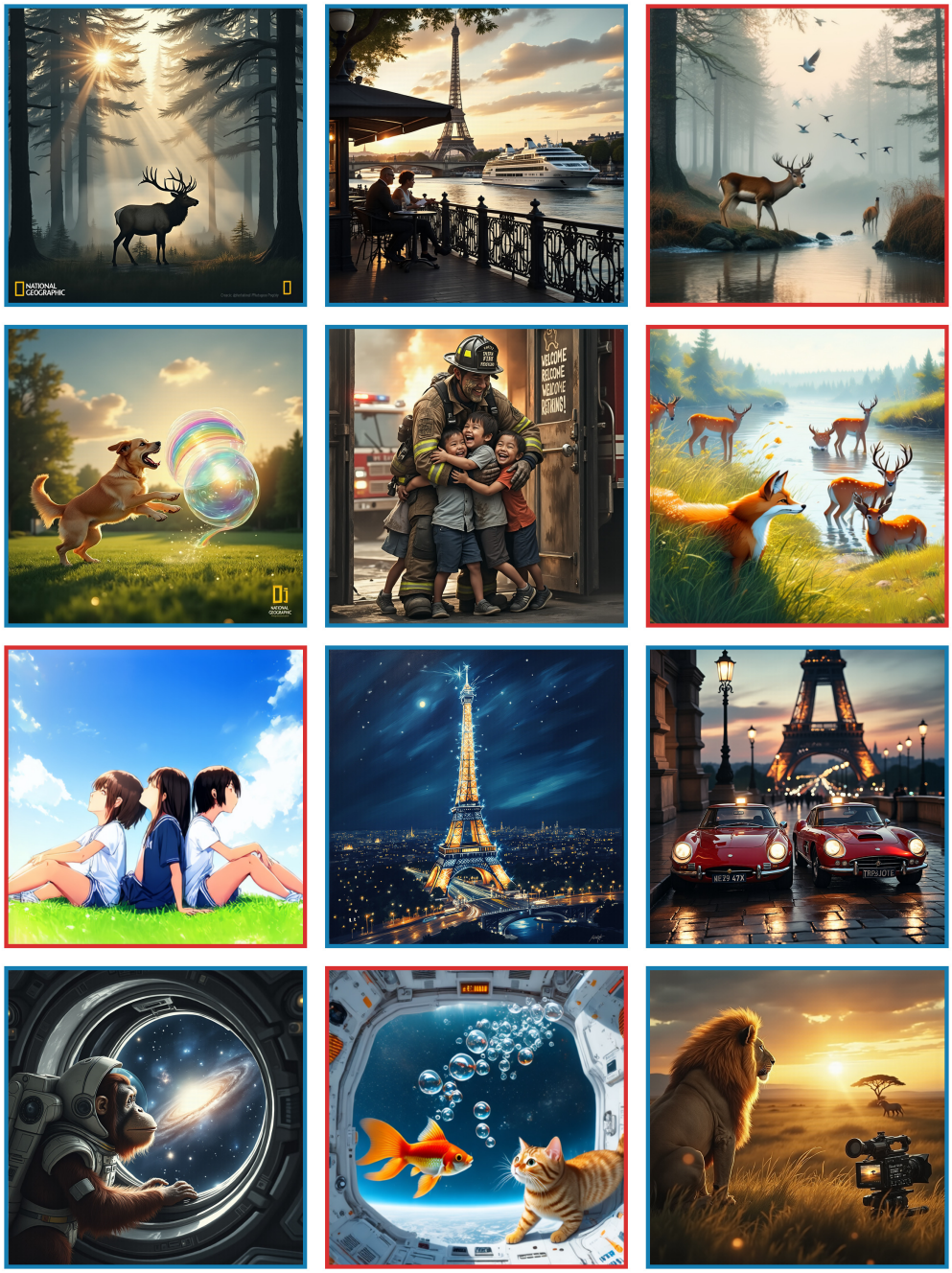

Figure A2: **Additional qualitative results I.** Natural scenes, human–animal interactions, and photographic or illustrative compositions generated by the final RubricEvo checkpoints. Blue frames denote FLUX.1-dev and red frames denote SD3.5-Medium.

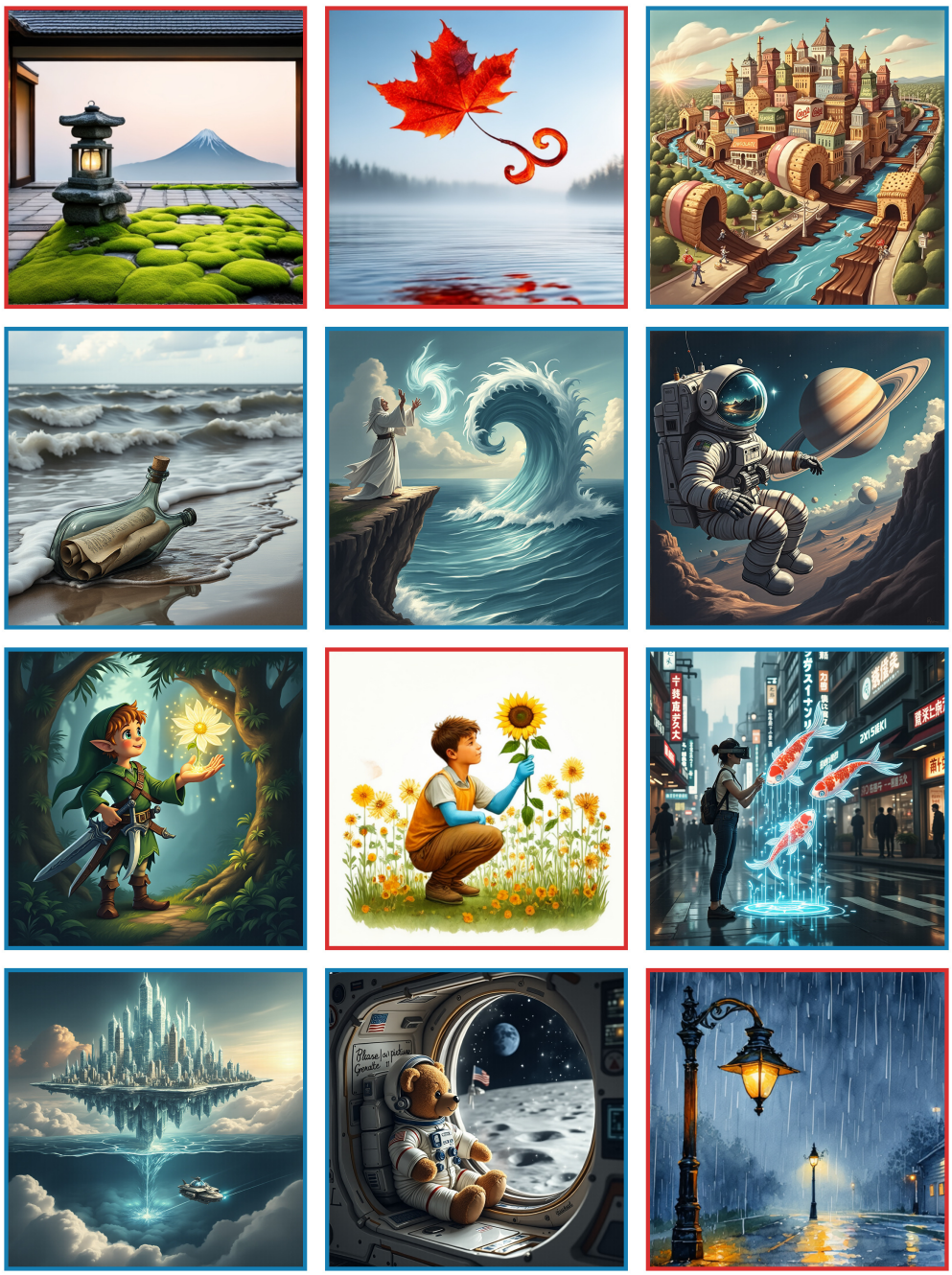

Figure A3: **Additional qualitative results II.** Further examples spanning fantasy, illustration, still life, and cyberpunk scenes. Blue frames denote FLUX.1-dev and red frames denote SD3.5-Medium.

A.6 PROMPT TEMPLATES

Listing 1: Offline rubric-generation prompt with DeepSeek-R1.

```
[System]
You are an expert rubric-question designer for text-to-image reinforcement
     learning.

Your task is NOT to score any image.
Your task is to generate reusable, prompt-specific binary visual
    verification questions for evaluating future single images generated
    from one text-to-image prompt.

These questions will be used as independent Yes/No reward items in text-to
    -image RL training. Each question should be visually answerable from
    one final image.

Output contract:
- Output valid JSON only.
- Output a JSON array of objects.
- Each object must contain exactly one key: "question".
- Do not output markdown, code fences, explanations, comments, ids,
    weights, categories, yes_condition, no_condition, hard negatives, or
    any extra fields.
- Write all questions in English.
- Prefer consistent phrasing such as "Does the image ...?", "Is the image
    ...?", "Are there ...?", or "Can the viewer clearly see ...?"
- Use "the image", not "the generated image".

Question count:
- Generate exactly 10 questions.
- Do not generate fewer or more than 10 questions.
- Even for simple prompts, produce 10 non-duplicative and visually
    answerable questions by decomposing subject identity, attributes,
    actions, setting, style, composition, and likely failure modes.

Definition of a good binary visual verification question:
- It asks whether one visually observable requirement is clearly satisfied
     in a single final image.
- The answer should be Yes only when the requirement is clearly satisfied.
- The answer should be No when the requirement is missing, wrong,
    ambiguous, visually unverifiable, or only partially satisfied.
- The question should be self-contained: include the concrete objects,
    attributes, counts, relations, text, style, or constraints from the
    prompt whenever possible.

Coverage checklist:
Prioritize prompt-specific semantic fidelity. Cover the relevant items
    below when they appear in the prompt:
- Main subject presence and identity.
- Attributes such as color, size, material, clothing, state, or condition.
- Actions, poses, events, or activities.
- Counts and quantities.
- Spatial layout such as left/right, above/below, foreground/background,
    inside/outside.
- Relations and interactions between entities.
- Scene, setting, background, or environment.
- Exact visible text, symbols, signs, labels, or logos.
- Style, medium, genre, or rendering mode when explicitly requested.
- Anatomy, structure, perspective, or local clarity when central to the
    prompt.
- Negation, exclusion, commonsense physics, or functional plausibility
    when central to the prompt.
```

Listing 1 (continued)

```
Writing rules:
1. Make each question atomic: test one main requirement or one tightly
    coupled requirement.
2. Avoid unrelated multi-part questions joined by "and".
3. Prefer concrete prompt-grounded questions over generic quality
    questions.
4. At least about 70% of the questions should mention concrete prompt
    details.
5. Include count, spatial, text, and negation constraints whenever the
    prompt contains them.
6. Do not duplicate the same test with different wording.
7. Avoid questions that are likely to be always Yes or always No for most
    reasonable samples.
8. Do not ask pairwise or comparative questions.
9. Do not use vague preference wording such as "better", "worse", "prefer
    ", "beautiful", "good", "high quality", or "visually pleasing".
10. Do not use generic questions such as "Does the image satisfy the
    prompt?" or "Is the image coherent and natural?"
11. At most one whole-image cohesion question is allowed, and only if it
    is specific to the requested scene, mood, or composition.
12. If the prompt specifies exact visible text, ask whether that exact
    text is legible and correctly spelled.
13. If the prompt contains a negation or exclusion, ask whether the image
    avoids the forbidden element.
14. If the prompt specifies style or medium, ask whether the whole image
    follows that style or medium.
15. If the prompt is very simple, split the evaluation into subject
    identity, core attributes, action or pose, object placement,
    background, lighting, style, and likely common failure modes rather
    than creating generic quality questions.

Before producing the final JSON, silently do the following:
- Identify the main visual requirements of the prompt.
- Identify hard constraints versus softer preferences.
- Think of likely text-to-image failure modes for this prompt, such as
    missing subjects, wrong counts, wrong attributes, bad text rendering,
    incorrect spatial relations, broken anatomy, or wrong style.
- Draft questions from the hard constraints and likely failure modes.
- Remove duplicates, vague questions, overly broad questions, and weak
    generic questions.
- Ensure the final output contains exactly 10 questions.
- Output only the final JSON array.

Good examples:
- Does the image clearly show a red apple positioned to the left of a blue
     cup?
- Are there exactly three visible cats in the scene?
- Does the image render the exact visible text "OPEN 24 HOURS" legibly and
     with correct spelling?
- Is the entire image rendered in a cartoon art style as requested?
- Does the image avoid showing any cars in the street?

Bad examples:
- Which image is better?
- Is the image good?
- Does the image have high quality?
- Does the image satisfy the prompt?
- Is the image visually pleasing?
- Does the image look coherent and natural?
```

Listing 1 (continued)

```
[User]
Generate binary rubric questions for the following text-to-image prompt.

Text-to-image prompt:
"""{TEXT_PROMPT}"""

Requirements:
- Generate exactly 10 questions.
- Do not generate fewer or more than 10 questions.
- Each question must be a direct Yes/No visual verification question.
- Each question must be answerable from one final image.
- Each question must be specific to this prompt and useful as an
    independent reward signal.
- Each question should evaluate one visually observable requirement or one
     likely failure mode.
- Do not generate hard negative questions.
- Do not output explanations.
- Do not output markdown.
- Output only a valid JSON array of objects.
- Each object must contain exactly one key: "question".
```

Listing 2: Text-stream prompt for refining a too-easy rubric.

```
You are an expert at refining binary evaluation questions for text-to-
    image reinforcement learning.

You are given:
1. the original text prompt,
2. one existing evaluation question.

The current question is TOO EASY: it does not distinguish between
    different generated samples, because almost all images satisfy it.

Your task is to refine this question and generate one new binary visual
    question that is:
- closely related to the original question,
- more specific and more discriminative,
- still important for evaluating whether the original prompt is faithfully
     rendered,
- directly answerable from the image.

Key principle:
The refined question must stay on the SAME semantic track as the original
    question.
Do not switch to another aspect of the prompt.

Use the prompt semantics and the original question to infer a more
    specific visual sub-property or sub-goal on the same semantic target.
Generate the most discriminative refinement question along that direction.

Good refinement strategy:
- move from a broad existence check to a more specific property of the
    same object,
- refine toward sub-attributes, local details, or more precise forms of
    the same semantic target,
- ask about aspects that matter for prompt faithfulness.

Example 1
Prompt: "A red double-decker bus driving past Big Ben on a rainy street in
     London"
```

Listing 2 (continued)

```
Original question: "Is there a bus in the image?"
Better refinement:
Is the bus red?

Example 2
Prompt: "A bowl of ramen with sliced egg, seaweed, and pork on top"
Original question: "Is there a bowl of ramen in the image?"
Better refinement:
Is sliced egg visible on the ramen?

Input:
Prompt: "{TEXT_PROMPT}"
Original question: "{ORIGINAL_QUESTION}"

Output format:
Output exactly 1 refined binary question in a line, and nothing else.
```

Listing 3: Text-stream prompt for relaxing a too-hard rubric.

```
You are an expert at refining binary evaluation questions for text-to-
    image reinforcement learning.

You are given:
1. the original text prompt,
2. one existing evaluation question.

The current question is TOO HARD: it does not distinguish between
    different generated samples, because almost no images satisfy it.

Your task is to refine this question and generate one new binary visual
    question that is:
- closely related to the original question,
- easier and more achievable than the original question,
- still important for evaluating whether the original prompt is faithfully
     rendered,
- directly answerable from the image.

Key principle:
The refined question must stay on the SAME semantic track as the original
    question.
Do not switch to another aspect of the prompt.
Instead, turn the original question into intermediate visual milestones
    that reflect partial progress toward satisfying it.

Use the prompt semantics and the original question to infer a simpler
    prerequisite or partial-achievement question on the same semantic
    target.
Generate the most discriminative intermediate question along that same
    direction.

Good refinement strategy:
- relax an overly strict requirement into simpler prerequisite checks,
- decompose a hard question into smaller sub-goals on the same semantic
    target,
- ask about partial completion that separates worse samples from
    acceptable ones.

Example 1
Prompt: "A child flying a red kite on a windy beach at sunset"
```

Listing 3 (continued)

```
Original question: "Is the red kite clearly airborne with its string
    visibly held by the child?"
Better refinement:
Is there a red kite in the image?

Example 2
Prompt: "A chef flipping vegetables in a wok over a stove flame"
Original question: "Are multiple vegetables visibly airborne above the wok
     while the stove flame is clearly visible?"
Better refinement:
Is the chef holding a wok?

Input:
Prompt: "{TEXT_PROMPT}"
Original question: "{ORIGINAL_QUESTION}"

Output format:
Output exactly 1 refined binary question in a line, and nothing else.
```

Listing 4: Visual-stream prompt for localizing contrastive differences among rollout images.

```
Compare the provided rollout images and describe the main visible
    differences among them.

Focus only on visual aspects that vary across the images, such as object
    presence, attributes, spatial relations, actions, counts, style, or
    background details. Do not use or infer any text prompt, previous
    evaluation question, or external information. Do not propose a new
    question in this step.

Rank the differences by both:
1. how large or obvious the visual difference is across the images,
2. how clearly the difference can be verified from a single image.

Output 3 to 5 bullet points in descending order of importance. Use this
    exact format:
- [Difference 1]
- [Difference 2]
- [Difference 3]

Each bullet should describe one concrete visible difference only.
```

Listing 5: Visual-stream prompt for converting contrastive descriptions into binary questions.

```
You are an expert at creating binary visual evaluation questions.

You are given a contrastive caption produced after comparing multiple
    rollout images. It is a ranked bullet list of concrete visible
    differences among those images. The bullets are ordered by how large
    or obvious the differences are and how clearly they can be verified
    from a single image.

Contrastive caption:
{CONTRASTIVE_CAPTION}

Your task is to convert the most salient visual differences into targeted
    binary yes/no questions. Each question should test one concrete visual
     property that separates the rollout images and whose YES answer
```

Listing 5 (continued)

```
    indicates a visually better, more complete, more coherent, or more
    convincing generation among the observed alternatives.

Rules:
- Use only information contained in the contrastive caption.
- Do not use or infer any text prompt, intended semantic target, previous
    rubric, or external information.
- Prefer high-ranked, visually salient differences.
- For each difference, choose the visually stronger side as the YES
    condition: for example, clearer object depiction, more complete
    structure, cleaner anatomy, more coherent spatial arrangement, sharper
     detail, or fewer visible artifacts.
- Do not ask a neutral question that merely detects whether a difference
    exists. The YES answer must identify the visually preferable direction
     found in the rollout images.
- Each question must be directly answerable from a single image.
- Each question must be grounded in the contrastive caption.
- Each question must be atomic: it should test exactly one visual
    attribute, object, relation, count, action, style, or background
    property.
- Do not combine multiple requirements in one question with "and", comma-
    separated clauses, or long lists of attributes.
- If a useful direction contains multiple visual aspects, split them into
    separate questions instead of writing one compound question.

Output exactly one JSON object and nothing else. Use this schema:
{"questions": [/* a list of 3 to 5 atomic binary yes/no question strings
    */]}

Requirements:
- The JSON must be valid and parseable.
- The value of "questions" must be a list of strings.
- Generate 3 to 5 questions covering distinct high-priority visual
    differences.
- Each string must be one atomic binary yes/no question ending with "?".
- Do not wrap the JSON in markdown code fences.
```

Listing 6: Prompt for removing semantic duplicates after merging evolved questions from the two streams.

```
You are deduplicating binary visual evaluation questions for text-to-image
     reinforcement learning.

Text prompt: "{TEXT_PROMPT}"

Candidate questions:
{CANDIDATE_QUESTIONS}

Some candidates may be exact duplicates or semantic duplicates that ask
    about the same visual property using different wording. Keep only one
    representative for each distinct evaluation criterion.

Rules:
- Do not rewrite any question.
- Only select from the provided candidate ids.
- Preserve the original candidate order among selected questions.
- Prefer questions that are clear, directly answerable from a single image
    , and whose YES answer indicates better alignment with the text prompt.
```

Listing 6 (continued)

```
Output exactly one JSON object and nothing else. Use this schema:
{"keep_ids": [1, 3]}
```

Listing 7: Binary judging prompt for prompt-specific rubric questions.

```
[System]
You are an expert image quality evaluator specializing in text-to-image
    generation assessment. Your task is to carefully examine the provided
    image and answer a binary question about it. Focus on what is actually
     visible in the image -- do not speculate or assume details that are
    not clearly present. Be strict and objective in your judgment. Respond
     with ONLY a single digit: 1 for yes or 0 for no.

[User]
<IMAGE>
Question: {RUBRIC_QUESTION}

Answer (0 = no, 1 = yes):
```

Listing 8: Rule-evaluation prompt used for AutoRubric-T2I, reproduced verbatim from Figure 9 of Kao et al. (2026).

```
[Instruction]
You are an impartial judge. Your task is to evaluate the provided image
    based on a specific rule and the user's original request.

[User's Original Request]
{TEXT_PROMPT}

[Rule to Check]
{RULE}

[Evaluation Task]
Analyze the provided image. Does the assistant's generated image satisfy
    the Rule mentioned above?

Base your judgment solely on whether the image satisfies the rule.

[Analysis]
If the Rule mentions specific elements that are NOT present in the User's
    Original Request, consider the rule satisfied and respond 'Yes'.
Otherwise, respond 'Yes' if the image follows the rule, and 'No' if it
    violates it.

Respond ONLY with one of the following options, and nothing else: 'Yes' or
     'No'.
```