VERIFIABLE VISUAL REWARDS TRANSFER FROM SYNTHETIC SCENES TO NATURAL PROMPTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Precise instruction following in image generation, such as satisfying object counts and spatial relations, is optimized using learned evaluators such as object detectors and vision-language models that are unreliable themselves. We introduce *Verifiable Visual Rewards* (VVR), the first framework for programmatically verifiable image rewards, reveal a capability gap in current models and show that training on it generalizes to natural prompts. Each VVR task is a scene of geometric objects and relations among them, from which we derive both the prompt and a deterministic verifier, so tasks can be generated in any number and at any chosen complexity. We release VVRBENCH, with 10,000 tasks over 46 constraint types, and VVRBENCH-Challenge, with 720 more complex tasks; the strongest model we evaluate—GPT-Image-2.5—solves 21.4% of VVRBENCH-Challenge. Using VVR scores as rewards for reinforcement learning (RLVVR) raises the accuracy of Stable Diffusion 3.5 Medium on VVRBENCH from 2.8% to 28.3% and demonstrates consistent easy-to-hard generalization. These gains extend to out-of-domain benchmarks, and mixing VVR into existing objectives further improves overall performance and human preference, motivating the adoption of VVR into standard image generation post-training recipes.

1 INTRODUCTION

Reinforcement learning with verifiable rewards has improved how precisely language models follow instructions: constraints such as *use the word X at least three times* are checked by code and used directly as rewards (Zhou et al., 2023; Lambert et al., 2025; Pyatkin et al., 2025), but image generation has no equivalent reward. Text-to-image generators often fail to follow instructions precisely: given *three red circles to the left of two blue squares*, they draw the wrong counts, colors, or positions, and fail more often as a prompt combines more requirements (Ghosh et al., 2023; Huang et al., 2023; Kamath et al., 2025). Post-training rewards for instruction following come from learned evaluators: preference models (Kirstain et al., 2023; Xu et al., 2023), vision-language models that answer questions about the image (Hu et al., 2023; Cho et al., 2024a; Lin et al., 2024), and object detectors (Ghosh et al., 2023). These evaluators make errors on the judgments that instruction following depends on (Saxon et al., 2024; Wiles et al., 2025; Kajić et al., 2024; Chen et al., 2025b; Kamath et al., 2025), and policies trained on them exploit these errors (Zhang et al., 2024; Kim et al., 2024; Hong et al., 2026).

We introduce *Verifiable Visual Rewards* (VVR), the first framework for programmatically verifiable image rewards, in which the generated image is scored deterministically by *verifiers*: Python functions over pixels, with no learned detector, OCR system, embedding model, or vision-language model. VVR covers instructions with clear, objective requirements combining color, count, shape, and spatial relations (Figure 1). Unlike constraints in text instruction following, which govern mostly separate properties of the output (e.g., length, keyword, format) and can be excluded pair by pair (Pyatkin et al., 2025), visual constraints lead to more complicated compatibility conflicts. For example, in *"A contains B, B contains C, and C contains A,"* every pair of constraints can be satisfiable, but the three together are not. Therefore, we propose a generator that guarantees constraint satisfiability under compositions, and compose natural-language instructions from the valid constraint sets. With our generator and constraint taxonomy, new VVR tasks can be generated in any number and at any chosen complexity, for evaluation or for training. Program verifiers of each VVR constraint also enable fine-grained diagnosis of generator capability over different types of instructions.

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.

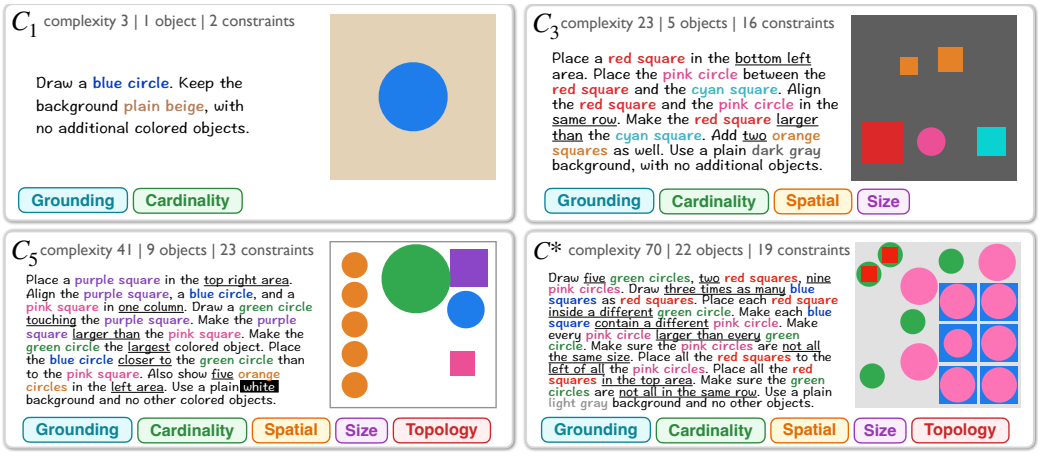

Figure 1: Example VVRBENCH tasks from different complexity ranges (C_1–C_5) and Challenge (C^*). Each panel shows the prompt, its complexity, the number of object instances, the number of constraints, the active constraint families, and a reference image that satisfies every constraint.

VVRBENCH. We instantiate VVR with colored geometric shapes and 46 constraint types over counts, attributes, and spatial relations and release **VVRBENCH**, with 10,000 tasks across five complexity ranges, and **VVRBENCH-Challenge**, with 720 more complex tasks to discriminate among frontier models (§3). The strongest open-weight model, FLUX.2-dev, achieves 19.2% accuracy on VVRBENCH, and the strongest model overall, GPT-Image-2.5-Sunburst, solves 21.4% of VVRBENCH-Challenge. Failures concentrate in the **Cardinality** (e.g., *"twice as many A as B"*) and **Topology** (e.g., *"each A is inside a different B"*) constraint families. The benchmark can be updated with higher complexity as frontier models evolve.

RLVVR. We use the verifiable VVR scores as rewards for reinforcement learning (RLVVR) to train image generators to follow instructions precisely. To study how training complexity affects generalization, we procedurally generate two training corpora: VVR-Easy contains only low complexity tasks of at most one constraint family, and VVR-Matched matches the VVRBENCH distribution. We show that 1) training on easy distribution generalizes to harder tasks, 2) training on harder tasks teaches compositionality, 3) training on colored shapes transfer to out-of-domain natural prompts to improve position and counting, and, most importantly, 4) mixing VVR with existing post-training objectives (e.g., GenEval2, OCR, PickScore) improves general benchmark performance and human preference, motivating the adoption of VVR into standard image generation post-training recipes. Our contributions are:

1. VVR, the first framework for programmatically verifiable image rewards, whose generator composes constraints on color, count, shape, and spatial relations into satisfiable instructions at any chosen complexity, each constraint checked by its own verifier (§2).
2. VVRBENCH and VVRBENCH-Challenge, benchmark that reveals capability gap of image generators to follow instructions, on which even frontier image generators fail most complex tasks, with failures tracable to specific constraint types (§3).
3. RLVVR, reinforcement learning with VVR rewards, which improves precise instruction following on tasks harder than those seen in training, transfers from synthetic scenes to natural prompts, and, mixed with existing post-training objectives, improves general benchmark performance and human preference (§4).

2 VERIFIABLE VISUAL REWARDS

2.1 VVR TASK REPRESENTATION

A VVR task specifies the requirements of a scene of colored shapes:

$$s = (\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F}, p). \tag{1}$$

Here $\mathcal{G}$ is a set of object groups, $\mathcal{B}$ is a set of background constraints, $\mathcal{A}$ is a set of active constraints on the object groups, $\mathcal{F}$ is a set of forbidden-content constraints, and p is the natural-language

Table 1: Constraint library. VVR groups 46 constraint types into five families. Appendix A.1 lists all types, and Appendix C.2 defines their verifiers.

Family	Visual property	Example constraint types
Grounding	Object identity and attribute binding	Color; shape; color–shape binding
Cardinality	Quantities and count comparisons	Exact count; equal, greater, or fewer counts; count ratios (X times as many)
Spatial	Position and arrangement	Image regions; relative order; alignment; grids; distance comparisons
Size	Relative visual extent	Pairwise and groupwise size; within-group variation; extrema
Topology	Contact and enclosure	Touching; separation; containment; distinct containment

instruction. In VVRBENCH, $\mathcal{B}$ and $\mathcal{F}$ are the same in every task: a plain background color and $\mathcal{F} = \{\texttt{no_unrequested_objects}\}$, so we focus the rest of the section on the constraints in $\mathcal{A}$.

Each constraint in $\mathcal{A}$ applies one of 46 constraint types from the constraint library (Table 1) to one or more object groups. For example, `exact_count(g1; 3)` is a unary constraint that requires group `g1` to contain three objects, and `left_of(g1,g2)` requires group `g1` to appear left of group `g2`. Every group has exactly one color constraint and one shape constraint; all other constraints are optional. Appendix A.2 shows a complete task.

A valid task must satisfy three desiderata: **1) Well-formed:** every constraint uses a defined type with supported parameter values, such as one of eight colors or three shapes, and refers to as many object groups in $\mathcal{G}$ as its type requires; **2) Jointly satisfiable:** at least one placement and sizing of the specified objects satisfies all constraints in $\mathcal{B}$, $\mathcal{A}$, and $\mathcal{F}$ simultaneously; and **3) Faithfully expressed:** p states every constraint in $\mathcal{B}$, $\mathcal{A}$, and $\mathcal{F}$ without any addition or omission.

2.2 TASK GENERATION

We now walk through the stages of the VVR generator that produces well-formed, jointly satisfiable, and faithfully expressed tasks. Appendix B provides an example generation and validation details.

1. It first creates a *scene* by sampling background constraints $\mathcal{B}$ and objects. It randomly assigns every object a color, shape, position, and size, forming object groups $\mathcal{G}$, and samples forbidden-content constraints $\mathcal{F}$ that no object in the scene violates.
2. For each constraint type in the library, it lists all object group tuples with size corresponding to the type's arity. The constraint type and its input tuple form an *instantiated constraint*.
3. All instantiated constraints that are true under the constructed scene, checked by program verifiers, form *satisfiable constraint set* $\mathcal{A}^*$. From $\mathcal{A}^*$, multiple valid active constraint sets $\mathcal{A}$ can be sampled such that $\mathcal{A} \subseteq \mathcal{A}^*$.
4. A template τ with phrasing variants transforms each task requirement into natural language, $p = \tau(\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F})$, forming $s = (\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F}, p)$.

Every constraint instantiates a library type on a tuple of sampled groups whose length equals the type's arity, so every task is **well-formed** by construction. The scene satisfies $\mathcal{B}$, $\mathcal{F}$, and every constraint in $\mathcal{A}^*$, so it satisfies the task formed with any $\mathcal{A} \subseteq \mathcal{A}^*$, making the task **jointly satisfiable**. Finally, in the template τ, every requirement of s has a fixed phrase in p, and every phrase in p comes from a requirement of s, guaranteeing **expression faithfulness**.

Structural complexity estimates task difficulty. We define the structural complexity of a task as $C(s) = \sum_{a \in \mathcal{A}} c(a; s)$, where $c(a; s)$ is the complexity contribution of one constraint a in task s. $c(a; s)$ follows a fixed rule for each constraint type and grows with the number of object instances that a evaluates in s. Appendix Table 7 gives the cost of every constraint type. Every task has one background-color constraint and the forbidden-content constraint `no_unrequested_objects`, so the constraints in $\mathcal{B}$ and $\mathcal{F}$ are excluded.

Datasets. Given a target distribution $\mathcal{T}$ over constraint families and structural complexity range, the generator can retain task candidates to fit $\mathcal{T}$. Thus datasets can be built to evaluate or learn specific constraint types at specified difficulty. VVR datasets used by this paper and their complexity distribution are listed in Table 2.

Table 2: Training and evaluation datasets produced by the VVR task generator.

Dataset	Use	Size	Complexity
VVRBENCH	evaluation	10,000	3–48
VVRBENCH-Fast	evaluation	820	3–44, 20 each
VVRBENCH-Challenge	evaluation	720	45–80, 20 each
VVR-Easy	training	100,000	≤ 20
VVR-Matched	training	100,000	$\sim$VVRBENCH

2.3 DETERMINISTIC, REFERENCE-FREE CONSTRAINT VERIFICATION

VVR is an open-ended image generation task, where any image that satisfies all constraints receives full credit, so the verifier has to be reference-free. It takes in the generated RGB image x and the formal constraints $(\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F})$ of the task and produces a correctness decision.

Object extraction from pixels. VVR first extracts candidate objects from the generated image using deterministic pixel-level operations. 1) It produces a binary mask for each supported color, with fixed hue and contrast thresholds. 2) Connected-component analysis assigns the same label to foreground pixels connected by a path of edge- or corner-adjacent pixels; each labeled region is a candidate object. 3) Fixed contour measurements score each candidate as a circle, square, or triangle based on its aspect ratio, bounding-box coverage, and convexity. 4) Candidates are then matched to object groups by the color and shape constraints of each group. Appendix C.1 describes the handling of ambiguous colors, irregular contours, fragmented objects, and blurred boundaries.

Constraint verifier library. The verifier library $\mathcal{V}$ contains one *program verifier* for each constraint type: a Python function that applies the type's requirement to the extracted objects. Verifier decisions use fixed comparisons of object counts, positions, extents, or boundary distances. For a constraint $a \in \mathcal{B} \cup \mathcal{A} \cup \mathcal{F}$, the verifier v_a returns a partial-credit score $q_a(x, s) \in [0, 1]$ and a pass-or-fail decision $d_a(x, s) \in \{0, 1\}$. For example, consider $a =$ `left_of(g1,g2)`. The program verifier compares the mean horizontal centroid coordinates of `g1` and `g2` with a separation margin:

```
def left_of(g1, g2, margin):
    g1x = np.mean([c.centroid[0] for c in g1])
    g2x = np.mean([c.centroid[0] for c in g2])
    delta = g2x - g1x
    partial = np.clip(delta / max(margin, 1), 0, 1)
    return partial, delta >= margin
```

The function returns the partial-credit score $q_a(x, s)$ and pass-fail decision $d_a(x, s)$. Appendix C.2 gives the verifier code for every constraint in an example task, Appendix C.4 provides verifier validation details.

Scores. A generated image succeeds only if every constraint in $\mathcal{B}$, $\mathcal{A}$, and $\mathcal{F}$ passes:

$$r_{\text{exact}}(x, s) = \prod_{a \in \mathcal{B} \cup \mathcal{A} \cup \mathcal{F}} d_a(x, s). \tag{2}$$

VVRBENCH accuracy is the mean of r_{exact} across tasks. For training, we design a dense reward $r_{\text{dense}} \in [0, 1]$ that gives partial credit through the verifier partial-credit scores:

$$r_{\text{dense}}(x, s) = \psi(x, s) \sum_{a \in \mathcal{B} \cup \mathcal{A} \cup \mathcal{F}} w_a \, q_a(x, s). \tag{3}$$

The weights w_a are fixed by constraint type, and $\psi(x, s) \in [0, 1]$ is a multiplicative penalty factor that prevents the model from exploiting any single easy-to-learn constraint while ignoring others (Zhang et al., 2024; Hong et al., 2026). Appendix C.3 provides more details on w_a and ψ.

3 VVRBENCH

Benchmark splits. We evaluate on three benchmark splits produced by the generator (Table 2). **1) VVRBENCH**, the main benchmark, contains 10,000 tasks of complexity 3 to 48, which we report in five ranges C_1 to C_5 of about 2,000 tasks each. **2) VVRBENCH-Fast** is an 820-task subset covering the same range with 20 tasks at each integer complexity, for evaluating paid image APIs at a twelfth of the generation cost. **3) VVRBENCH-Challenge** contains 720 tasks of complexity 45 to 80, above the VVRBENCH range, to separate the strongest generators.

Models. We evaluate ten open-weight models: FLUX.2-dev (Black Forest Labs, 2025), HunyuanImage-2.1 (Tencent Hunyuan Team, 2025), Qwen-Image-2512 (Wu et al., 2025; Qwen Team, 2025), HiDream-I1-Full (Cai et al., 2025), FLUX.1-dev and FLUX.1-schnell (Black Forest Labs, 2024), Stable Diffusion 3.5 Medium and Large (Esser et al., 2024; Stability AI, 2024), SDXL (Podell et al., 2024), and Sana 1.6B (Xie et al., 2025), and seven API based models: GPT-Image-2.5-Sunburst (OpenAI, 2026b;c), GPT-Image-2 (OpenAI, 2026a), GPT-Image-1-mini (OpenAI, 2025), Gemini-3.1-Flash-Image (Google, 2026), Gemini-3.1-Flash-Lite-Image (Google DeepMind, 2026), Gemini-3-Pro-Image (Google DeepMind, 2025), and Gemini-2.5-Flash-Image (Google, 2025). Appendix D.1 gives the additional evaluation details.

Table 3: Accuracy (%) on the 10,000 VVRBENCH tasks, overall and by complexity range. C_1 to C_5 split the tasks by structural complexity $C(s)$ into five ranges of about 2,000 tasks each: 3–16, 16–21, 21–26, 26–31, and 31–48. Accuracy generally falls with complexity. GPT-Image-2 drops from 97.65% in C_1 to 65.72% in C_5, and no open weight model exceeds 20% overall.

Model	Accuracy (%) ↑	C_1	C_2	C_3	C_4	C_5
GPT-Image-2	**86.86**$_{\pm 0.68}$	**97.65**$_{\pm 0.74}$	**98.18**$_{\pm 0.70}$	**91.00**$_{\pm 1.34}$	**81.83**$_{\pm 1.75}$	**65.72**$_{\pm 2.10}$
GPT-Image-1-mini	26.40$_{\pm 0.87}$	74.06$_{\pm 1.93}$	36.78$_{\pm 2.18}$	12.47$_{\pm 1.53}$	4.70$_{\pm 1.02}$	2.39$_{\pm 0.76}$
FLUX.2-dev	19.15$_{\pm 0.78}$	49.86$_{\pm 2.15}$	24.26$_{\pm 1.96}$	12.42$_{\pm 1.52}$	5.91$_{\pm 1.12}$	2.24$_{\pm 0.74}$
HunyuanImage-2.1	18.79$_{\pm 0.78}$	45.53$_{\pm 2.15}$	19.64$_{\pm 1.83}$	14.84$_{\pm 1.63}$	8.61$_{\pm 1.31}$	4.29$_{\pm 0.98}$
Qwen-Image-2512	5.79$_{\pm 0.47}$	19.16$_{\pm 1.75}$	5.87$_{\pm 1.14}$	2.11$_{\pm 0.73}$	1.10$_{\pm 0.56}$	0.15$_{\pm 0.29}$
HiDream-I1-Full	4.22$_{\pm 0.41}$	16.47$_{\pm 1.65}$	3.06$_{\pm 0.87}$	0.80$_{\pm 0.50}$	0.20$_{\pm 0.31}$	0.00$_{\pm 0.19}$
FLUX.1-dev	3.87$_{\pm 0.40}$	15.51$_{\pm 1.62}$	2.18$_{\pm 0.75}$	0.96$_{\pm 0.53}$	0.15$_{\pm 0.29}$	0.00$_{\pm 0.19}$
FLUX.1-schnell	2.88$_{\pm 0.35}$	11.96$_{\pm 1.46}$	1.66$_{\pm 0.67}$	0.25$_{\pm 0.34}$	0.10$_{\pm 0.26}$	0.00$_{\pm 0.19}$
SD3.5 Medium	2.81$_{\pm 0.34}$	12.01$_{\pm 1.47}$	1.19$_{\pm 0.59}$	0.35$_{\pm 0.37}$	0.05$_{\pm 0.23}$	0.00$_{\pm 0.19}$
SD3.5 Large	2.45$_{\pm 0.32}$	10.47$_{\pm 1.39}$	1.14$_{\pm 0.58}$	0.20$_{\pm 0.32}$	0.05$_{\pm 0.23}$	0.00$_{\pm 0.19}$
SDXL 1.0	0.02$_{\pm 0.05}$	0.10$_{\pm 0.25}$	0.00$_{\pm 0.20}$	0.00$_{\pm 0.19}$	0.00$_{\pm 0.19}$	0.00$_{\pm 0.19}$
Sana 1.6B	0.00$_{\pm 0.04}$	0.00$_{\pm 0.18}$	0.00$_{\pm 0.20}$	0.00$_{\pm 0.19}$	0.00$_{\pm 0.19}$	0.00$_{\pm 0.19}$

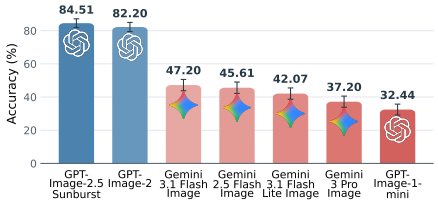

Figure 2: Accuracy of API models on VVRBENCH-Fast.

Table 4: Accuracy (%) on the 720 VVRBENCH-Challenge tasks. The best model solves 21.39% overall and 7.92% at complexity 69 to 80.

Model	Accuracy ↑	45 to 56	57 to 68	69 to 80
GPT-Image-2.5-Sunburst	**21.39**$_{\pm 3.14}$	**31.67**$_{\pm 6.13}$	**24.58**$_{\pm 5.82}$	**7.92**$_{\pm 4.12}$
GPT-Image-2	10.28$_{\pm 2.43}$	17.50$_{\pm 5.31}$	10.83$_{\pm 4.57}$	2.50$_{\pm 2.85}$
Gemini-3.1-Flash-Lite-Image	7.36$_{\pm 2.14}$	10.00$_{\pm 4.45}$	4.17$_{\pm 3.33}$	**7.92**$_{\pm 4.12}$
Gemini-3-Pro-Image	4.58$_{\pm 1.78}$	7.08$_{\pm 3.97}$	4.17$_{\pm 3.33}$	2.50$_{\pm 2.85}$
Gemini-3.1-Flash-Image	3.89$_{\pm 1.67}$	3.33$_{\pm 3.11}$	3.75$_{\pm 3.22}$	4.58$_{\pm 3.44}$
Gemini-2.5-Flash-Image	1.11$_{\pm 1.07}$	2.50$_{\pm 2.85}$	0.42$_{\pm 1.91}$	0.42$_{\pm 1.91}$
GPT-Image-1-mini	0.28$_{\pm 0.73}$	0.83$_{\pm 2.15}$	0.00$_{\pm 1.58}$	0.00$_{\pm 1.58}$

3.1 PRECISE INSTRUCTION FOLLOWING IS FAR FROM SOLVED

The strongest open-weight model, FLUX.2-dev, solves 19.15% of VVRBENCH tasks and 2.24% of C_5, and every other open-weight model solves less than 19% (Table 3). GPT-Image-2 solves 86.86% of all tasks, but its accuracy falls from 97.65% in C_1 to 65.72% in C_5. On VVRBENCH-Fast (Figure 2), GPT-Image-2.5-Sunburst and GPT-Image-2 solve 84.51% and 82.20% of the tasks, respectively, leading other API models by a large margin (exact scores in Appendix D.2).

VVRBENCH-Challenge separates frontier models. GPT-Image-2.5-Sunburst and GPT-Image-2 solve most VVRBENCH-Fast tasks and differ by only 2.3 points, so VVRBENCH barely separates the strongest generators. VVRBENCH-Challenge, sampled from a higher complexity distribution, discriminates among them and exposes new failure modes (Table 4). GPT-Image-2.5-Sunburst solves 21.39% of Challenge tasks, twice the 10.28% of GPT-Image-2, and its accuracy falls from 31.67% at complexity 45–56 to 24.58% at 57–68 and 7.92% at 69–80. Every other model solves at most 8% of VVRBENCH-Challenge, suggesting that there is still large room for improvement. Interestingly, we observe occasional abstention behaviors from all Gemini models, stating the instruction is unsatisfiable, demonstrating failure in spatial reasoning (Appendix D.3).

The uniform drop of model accuracy across increasing complexity bins validates the design of the structural complexity score as a model-independent heuristic to generate tasks with controlled difficulty. Appendix B.4 provides more details on complexity as a predictor of failure.

Finding 1. Open-weight generators fail most VVRBENCH tasks, and even the strongest API models lose accuracy sharply as complexity grows.

3.2 FAILURES CONCENTRATE IN COUNTING AND OBJECT MATCHING

The binary pass-fail score (Eq. 2) is composed of individual verifier decisions from each of the active constraints in each task. Figure 3 presents the constraint-level pass rate of the API models on VVRBENCH-Challenge. Organized by constraint families (Table 1), 95% of **Grounding** constraints are satisfied, while only 56% of **Topology** constraints are rendered, averaged across models. The

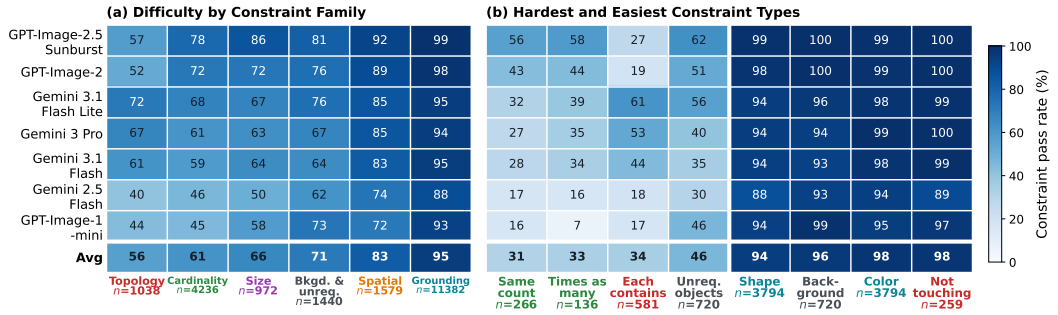

Figure 3: Pass rates of individual constraint for API models on VVRBENCH-Challenge, by family (left) and for the four constraint types with the lowest and the four with the highest average pass rates among those with at least 100 checks per model (right).

hardest constraint types concern counts or relations across object groups: **same count** passes in 31% of checks, **times as many** in 33%, and **each contains**, which requires each object of one group to contain a different object of another group, in 34%.

Appendix D.4 gives the pass rate of every constraint type, Appendix D.5 correlates each family to accuracy across all models at matched complexity, and Appendix D.6 shows typical failures in which a model adds objects that the prompt excludes.

Finding 2. Generators satisfy requirements on individual objects and pairs but fail requirements that constrain whole sets of objects.

4 RLVVR: VVR FOR DIFFUSION POST-TRAINING

VVR scores images with program verifiers, avoiding error propagation from unreliable learned evaluators, and is not limited to fixed prompt sets, so training can be scaled to any desired data size and difficulty distributions. These properties allow us to improve image generation instruction following by using VVR as a reward in reinforcement learning (RLVVR). In this section, we post-train image generators with RLVVR to answer the following research questions:

RQ1. Does RLVVR teach precise instruction following, and how does the complexity of the training tasks shape what is learned?

RQ2. Do the skills learned from synthetic scenes transfer to natural prompts beyond VVR?

RQ3. Is supervision from synthetic scenes complementary to existing post-training rewards?

4.1 EXPERIMENTAL SETUP

Data. We generate two training corpora using the VVR generator (§2.2). **VVR-Easy** contains tasks that contain at most one constraint family and have complexity of at most 20, and **VVR-Matched** matches the VVRBENCH distribution (complexity 3–48). Each dataset contains 100K VVR tasks after decontamination from benchmark data (Table 2). For reward-mixture experiments, we train with GenEval2 (Kamath et al., 2025), OCR (Liu et al., 2025a), and a five-reward objective that combines GenEval, GenEval2, OCR, PickScore (Kirstain et al., 2023), and UnifiedReward (Wang et al., 2025). Each of these objectives is trained alone and mixed with VVR-Easy, with equal number of prompts per objective.

Training. We train Stable Diffusion 3.5 Medium (Esser et al., 2024; Stability AI, 2024) with Flow-GRPO (Liu et al., 2025a), scoring each rollout with the VVR training reward r_{dense} (Section 2.3). We name each trained model after its training data. Appendix E reports training details.

Evaluation. We evaluate trained models on VVRBENCH, GenEval, GenEval2, OCR, PickScore, HPSv2.1, CLIPScore, aesthetic score, ImageReward, HPSv3, and UnifiedReward (Appendix E.1).

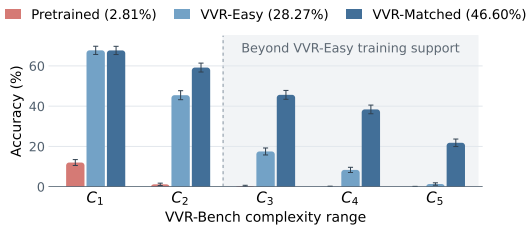

Figure 4: VVRBENCH accuracy by complexity range. Shaded ranges lie above the complexity of every VVR-Easy training task, and VVR-Easy improves them. Training on harder generated tasks (VVR-Matched) closes more of the gap.

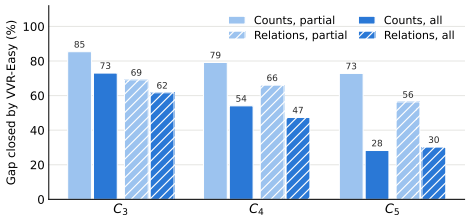

Figure 5: VVR-Easy closes the gap between pretrained and VVR-Match more effectively on ▫ partial scores (individual constraint) than ■ all-satisfy scores (compositionality) on both *count* and *relation* constraints.

Table 5: RLVVR with VVR-Easy and reward mixtures transfers to most benchmarks and metrics.

Training reward	VVR	GenEval	GenEval2	OCR	PickScore	HPSv2.1	HPSv3	CLIPScore	Aesthetic	ImageReward	UnifiedReward
Pretrained	0.028	0.616	0.237	0.476	0.841	0.300	7.689	0.956	5.517	0.929	0.636
+ VVR-Easy	0.283	0.729	0.268	0.587	0.849	0.294	8.275	0.979	5.483	1.114	0.641
Δ	+0.255	+0.113	+0.031	+0.111	+0.008	−0.006	+0.586	+0.023	−0.034	+0.185	+0.005
GenEval2	0.039	0.688	0.454	0.501	0.848	0.297	8.289	0.974	5.523	1.110	0.635
+ VVR-Easy	0.218	0.718	0.478	0.532	0.848	0.302	8.511	0.976	5.533	1.155	0.637
Δ	+0.180	+0.030	+0.025	+0.030	+0.0004	+0.005	+0.222	+0.002	+0.009	+0.045	+0.0019
OCR	0.036	0.625	0.225	0.962	0.844	0.290	7.601	0.960	5.477	0.994	0.633
+ VVR-Easy	0.247	0.678	0.252	0.941	0.846	0.290	7.765	0.969	5.488	1.087	0.636
Δ	+0.212	+0.053	+0.027	−0.021	+0.0018	0.000	+0.164	+0.009	+0.012	+0.092	+0.0024
Five-reward	0.048	0.739	0.342	0.856	0.850	0.294	8.137	0.974	5.510	1.125	0.640
+ VVR-Easy	0.158	0.751	0.383	0.826	0.848	0.300	8.444	0.976	5.541	1.152	0.640
Δ	+0.110	+0.012	+0.041	−0.030	−0.0024†	+0.006	+0.307	+0.002	+0.031	+0.027	+0.0003

4.2 RQ1: RLVVR TEACHES PRECISE INSTRUCTION FOLLOWING

Training on VVR-Easy raises VVRBENCH accuracy from 2.81% to 28.27% (Figure 4). Every task in C_3–C_5 is more complex than any VVR-Easy task, and on these ranges accuracy still rises by 17.16, 8.31, and 1.35 points. Training on the benchmark distribution with VVR-Matched raises accuracy to 46.60% overall and to 45.62%, 38.39%, and 21.82% on C_3–C_5 (Appendix F.1).

We separate how reliably a model satisfies individual constraints from how well it satisfies compositional requirements, using the two kinds of constraint that nearly every complex task contains: counts and relations between groups (Appendix F.2). We compare a model's partial scores $q_a(x, s)$ on these constraints with how often it satisfies every count or every relation in a task. On tasks outside of its training complexity range, VVR-Easy closes 71% of the gap between the pretrained model and VVR-Matched in the partial scores of counts and relations, but only 51% in how often all counts or relations are satisfied (Figure 5). Easy tasks thus make individual constraints reliable, and satisfying many constraints in the same image is learned from large scenes.

Finding 3. Training only on easy tasks makes individual constraints reliable, including on harder tasks, and training on large scenes teaches compositionality.

4.3 RQ2: SKILLS LEARNED FROM SYNTHETIC SCENES TRANSFER TO NATURAL PROMPTS

Trained only on colored shapes, VVR-Easy improves the pretrained model on eight of ten non-VVR metrics, including GenEval by 0.113 and OCR by 0.111 (Table 5). Human annotators confirm the transfer: VVR-Easy is preferred by human annotators over the pretrained SD-3.5-M on their generations from 160 natural prompts outside VVR with a win rate of 71.6% (Table 6). Averaged over pairs of the three annotators, two annotators agree on 83.8% of the image pairs where both chose an image. Appendix G reports annotation details.

VVR-Easy also scores higher on 6 out of 9 natural prompt benchmarks than the model trained with the GenEval2 reward, whose training prompts name real objects—especially GenEval (0.729 vs. 0.688) and OCR (0.587 vs. 0.501). Notably, the GenEval gain comes from position (+0.150),

Table 6: Human preference win-rate (%) for the RLVVR-trained model over its baseline.

VVR win rate vs. baseline	VVR	GenEval2	GenEval	OCR	DrawBench	Outside VVR
VVR-Easy vs. pretrained	93.3 [86.7, 98.3]	78.8 [67.5, 88.8]	62.9 [50.4, 75.0]	69.6 [58.3, 80.4]	75.0 [65.4, 84.2]	71.6 [65.9, 77.1]
GenEval2 + VVR-Easy vs. GenEval2	89.2 [81.7, 95.4]	62.9 [50.8, 74.6]	57.5 [45.0, 69.6]	56.7 [44.6, 68.8]	57.5 [45.4, 69.2]	58.6 [52.6, 64.6]

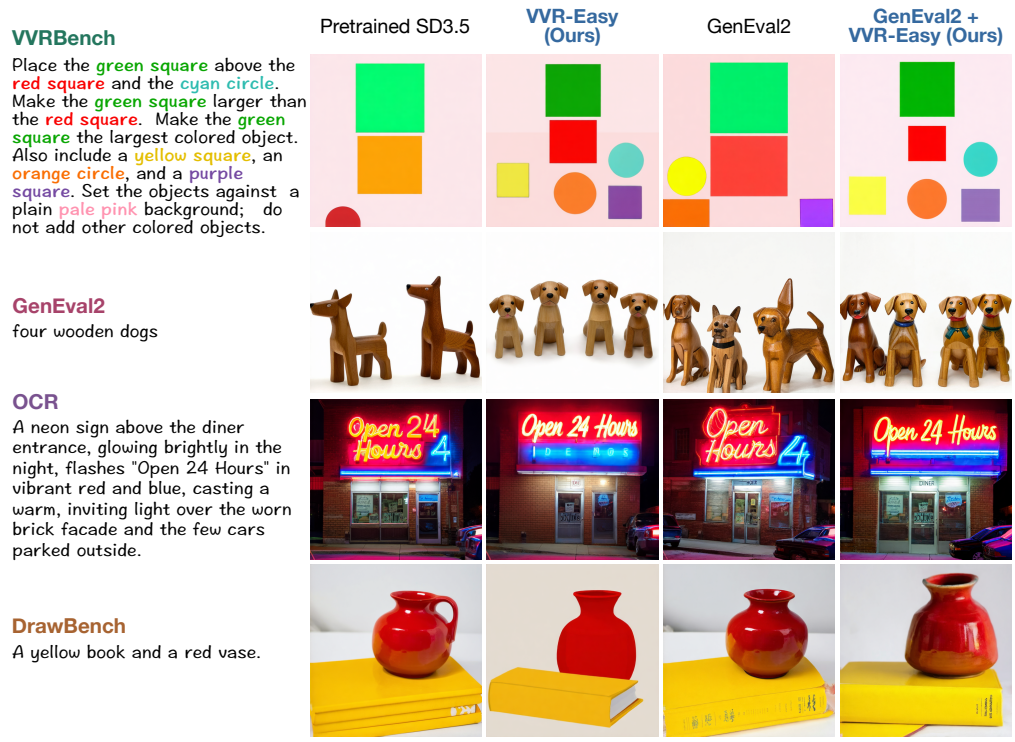

Figure 6: Final SD3.5-M models generated from the same prompt and seed within each row. On the VVR prompt, the verifier accepts both VVR-trained outputs and rejects both baselines. On the DrawBench prompt, VVR-Easy renders the vase as a flat shape without shading, consistent with its lower aesthetic and HPSv2.1 scores; GenEval2 with VVR-Easy retains shading.

counting (+0.103), and color attribution (+0.025), all skills that VVR trains, while the single object, two object, and colors categories are comparable to the GenEval2-trained model.

Finding 4. Skills learned from synthetic VVR scenes transfer to natural prompts, especially in position and counting.

4.4 RQ3: Supervision from synthetic scenes complements existing rewards

Mixed with GenEval2, VVR-Easy raises all ten non-VVR metrics in Table 5, including GenEval2 itself (+0.025). The largest metric gains are in GenEval (+0.030), OCR (+0.030), HPSv3 (+0.222), and ImageReward (+0.045). Human annotators prefer the mixture to GenEval2 alone on the 160 prompts outside VVR with a win rate of 58.6% (Table 6).

Mixed with OCR and with the five-reward objective, VVR-Easy raises eight of ten metrics each, with the largest gains in GenEval by 0.053 and ImageReward by 0.092 in the OCR mixture, and GenEval2 by 0.041 and HPSv3 by 0.307 in the five-reward mixture. In these two mixtures, native OCR accuracy falls by 0.021 and 0.030, and in the five-reward mixture PickScore falls by 0.002, since each mixture trains on fewer prompts from the original sources. Appendix F.3 reports all conditions and intervals.

Finding 5. Adding VVR tasks to existing post-training objectives improves human preference and most non-VVR metrics.

5 Related Work

Verifiable rewards. Language-model post-training uses executable rules to score answers and instruction constraints (Lambert et al., 2025; Guo et al., 2025; Zhou et al., 2023). Procedural environments extend this idea by generating tasks of controlled difficulty together with exact verifiers (Stojanovski et al., 2025; Liu et al., 2025b; Chen et al., 2025a). CLEVR derives questions and their answers from structured scenes to evaluate visual question answering (Johnson et al., 2017). VVR applies the same scene-first construction to image generation: each task produces the instruction, its verifier, and its complexity score. When the model outputs a program, such as SVG or TikZ, rule-based rewards can read object geometry from the program itself (Li et al., 2026; Belouadi et al., 2024), or compare its rendering with a reference image (Rodriguez et al., 2025). A text-to-image model outputs only pixels, so a verifier must recover each object, its color, shape, and extent from the raster, and must accept every image that satisfies the instruction rather than a single reference. VVR performs this recovery with fixed image operations.

Rewards for text-to-image post-training. Diffusion and flow models are post-trained with policy gradients (Black et al., 2024; Fan et al., 2023), differentiable rewards (Xu et al., 2023; Clark et al., 2024), preference optimization (Wallace et al., 2024), and online reinforcement learning for flow models (Liu et al., 2025a; Xue et al., 2025). Instruction-conditioned rewards commonly use learned preference models (Kirstain et al., 2023; Xu et al., 2023; Wu et al., 2023; Wang et al., 2025) or deterministic rules applied to learned perception. For example, Flow GRPO scores GenEval detections and OCR outputs (Liu et al., 2025a; Ghosh et al., 2023), SpatialReward combines detectors with a vision-language model (Zhou et al., 2026), and AlphaGRPO answers decomposed questions with a multimodal model (Huang et al., 2026). Such learned signals can be overoptimized (Zhang et al., 2024). DDPO also studies JPEG compressibility, a model-free image statistic that does not evaluate the instruction (Black et al., 2024). VVR provides an instruction-conditioned, model-free reward computed directly from generated pixels and can be mixed with these existing objectives.

Text-to-image evaluation. Text-to-image evaluation commonly uses embedding and question-answering metrics (Hessel et al., 2021; Hu et al., 2023; Cho et al., 2024a; Lin et al., 2024) and benchmarks for prompt alignment and composition (Saharia et al., 2022; Yu et al., 2022; Ghosh et al., 2023; Huang et al., 2023; Hu et al., 2024) that score open prompts with learned models. GenEval2 replaces the GenEval detector with a vision-language judge after detector-based scores diverged from human judgments on stronger generators (Kamath et al., 2025). Diagnostic benchmarks can generate prompts from synthetic scene configurations, but still score the outputs with detectors or vision-language models (Cho et al., 2024b; Wu et al., 2024). Each VVR task instead defines a reference-free pixel verifier that accepts any image satisfying the requested constraints. VVR uses these verifiers both for evaluation and as a post-training reward.

6 Conclusion

In this paper, we introduce Verifiable Visual Rewards (VVR), where open-ended image generation can be scored by deterministic program verifiers to provide both evaluation feedback and post-training signals. VVR tasks can be generated procedurally given any target distribution over constraint types and complexity levels. We release VVRBench, 10K verifiable image generation tasks where the model is asked to draw geometric objects with specified color, shape, count, and spatial relations, and show that models struggle with visual instruction following. A VVRBench-Challenge set where the strongest frontier image generation model, GPT-Image-2.5-Sunburst, solves only 21.4% of the tasks. We then train image generators with VVR scores as an RL reward (RLVVR) significantly improves instruction following both on VVR tasks and on natural prompts unseen during training. Mixing VVR with existing post-training objectives for image generation, such as GenEval2, leads to further gains on a broad evaluation suite and human preference, motivating its adoption into standard post-training recipes.

LIMITATIONS AND FUTURE DIRECTIONS

VVRBENCH uses eight colors, three shapes, and plain backgrounds; future work can add more shapes, textures, and object types as new program verifiers. VVR currently covers 2D geometric objects, and future work can extend it to 3D renderings or 2D projections of 3D objects.

VVR is constrained to text-to-image generation; the same constraints could be applied to image editing. New constraints such as motion, velocity, acceleration, are also convertible to program verifiers and can be applied to video generation. VVR outputs with their verifier decisions could be used to evaluate or train learned reward models and VLM judges.

We post-train SD3.5-M with Flow-GRPO; applying RLVVR to other image generators and RL algorithms is left to future work. RLVVR is an RL-Zero recipe: we apply Flow-GRPO directly to the pretrained SD3.5-M. Mid-training on VVR data with supervised fine-tuning or DPO before the RL stage, with different data mixtures, could further improve instruction following in image generation.

RLVVR uses the combined dense reward r_{dense}, but the program verifiers also report which constraints fail. This feedback allows a range of reward designs, for example weighting constraint families differently according to the desired model behavior. The complexity-controlled task generator also allows adaptive curricula for RLVVR.

Several API models incorrectly decline some VVR tasks as contradictory, although every task is satisfiable. Our analysis is constrained to case studies due to the small number of abstentions, but VVR tasks can be used to evaluate, and further train for, correct abstention decisions in image generators, VLM, and even LLMs to improve spatial reasoning.

AI USE STATEMENT

Generative AI tools were used to assist with code navigation, debugging, analysis scripting, and manuscript polishing. The authors take responsibility for the final content.

ETHICS STATEMENT

The annotation in this paper labels generated images of synthetic scenes and public benchmark prompts and involves no personal or sensitive data.

REPRODUCIBILITY STATEMENT

We release all three benchmark splits, the two training corpora, the verifier, the task generator, and the scripts that build every table and figure, together with evaluation prompts, training configurations, and model checkpoints. The appendix records reward formulas, full results tables, and dataset statistics.

REFERENCES

Jonas Belouadi, Simone Paolo Ponzetto, and Steffen Eger. DeTikZify: Synthesizing Graphics Programs for Scientific Figures and Sketches with TikZ. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2405.15306.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2305.13301.

Black Forest Labs. FLUX. Black Forest Labs GitHub repository, 2024. URL https://github.com/black-forest-labs/flux. FLUX.1 [dev] and [schnell]; citation as given in the official repository.

Black Forest Labs. FLUX.2: Frontier visual intelligence. Black Forest Labs blog post, 2025. URL https://bfl.ai/blog/flux-2. Blog post, November 25, 2025; citation as given in github.com/black-forest-labs/flux2.

Qi Cai, Jingwen Chen, Yang Chen, Yehao Li, Fuchen Long, Yingwei Pan, Zhaofan Qiu, Yiheng Zhang, Fengbin Gao, Peihan Xu, Yimeng Wang, Kai Yu, Wenxuan Chen, Ziwei Feng, Zijian Gong, Jianzhuang Pan, Yi Peng, Rui Tian, Siyu Wang, Bo Zhao, Ting Yao, and Tao Mei. HiDream-I1: A high-efficient image generative foundation model with sparse diffusion transformer, 2025. URL https://arxiv.org/abs/2505.22705.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a. URL https://arxiv.org/abs/2505.19914. arXiv:2505.19914.

Zhaorun Chen, Yichao Du, Zichen Wen, Yiyang Zhou, Chenhang Cui, Zhenzhen Weng, Haoqin Tu, Chaoqi Wang, Zhengwei Tong, Qinglan Huang, Canyu Chen, Qinghao Ye, Zhihong Zhu, Yuqing Zhang, Jiawei Zhou, Zhuokai Zhao, Rafael Rafailov, Chelsea Finn, and Huaxiu Yao. MJ-Bench: Is your multimodal reward model really a good judge for text-to-image generation? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025b. URL https://arxiv.org/abs/2407.04842.

Jaemin Cho, Yushi Hu, Roopal Garg, Peter Anderson, Ranjay Krishna, Jason Baldridge, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *International Conference on Learning Representations (ICLR)*, 2024a. URL https://arxiv.org/abs/2310.18235.

Jaemin Cho, Linjie Li, Zhengyuan Yang, Zhe Gan, Lijuan Wang, and Mohit Bansal. Diagnostic benchmark and iterative inpainting for layout-guided image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024b. URL https://arxiv.org/abs/2304.06671.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.17400.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024. URL https://arxiv.org/abs/2403.03206.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.16381.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2310.11513.

Google. Introducing Gemini 2.5 Flash Image, our state-of-the-art image model. Google Developers Blog, 2025. URL https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/. Google Developers Blog, August 26, 2025 ("Nano Banana"); model card: https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Model-Card.pdf.

Google. Nano Banana 2: Google's latest AI image generation model. Google blog post, 2026. URL https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/. Gemini 3.1 Flash Image, launched February 26, 2026; model page https://deepmind.google/models/gemini-image/flash/.

Google DeepMind. Gemini 3 Pro Image model card. Google DeepMind model card, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Image-Model-Card.pdf. "Nano Banana Pro"; published November 2025.

Google DeepMind. Gemini 3.1 Flash-Lite Image (Nano Banana 2 Lite). Google DeepMind model page, 2026. URL https://deepmind.google/models/gemini-image/flash-lite/. Model ID gemini-3.1-flash-lite-image, released June 30, 2026 (Gemini API release notes); https://ai.google.dev/gemini-api/docs/models/gemini-3.1-flash-lite-image.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z. arXiv:2501.12948 ("DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning").

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 (EMNLP)*, 2021. URL https://arxiv.org/abs/2104.08718.

Yunqi Hong, Kuei-Chun Kao, Hengguang Zhou, and Cho-Jui Hsieh. Understanding Reward Hacking in Text-to-Image Reinforcement Learning. *arXiv preprint arXiv:2601.03468*, 2026. URL https://arxiv.org/abs/2601.03468.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip diffusion models with LLM for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024. URL https://arxiv.org/abs/2403.05135.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. TIFA: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2303.11897.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench: A comprehensive benchmark for open-world compositional text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2307.06350v2. arXiv:2307.06350v2.

Runhui Huang, Jie Wu, Rui Yang, Zhe Liu, and Hengshuang Zhao. AlphaGRPO: Unlocking self-reflective multimodal generation in UMMs via decompositional verifiable reward. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2605.12495.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://arxiv.org/abs/1612.06890.

Ivana Kajić, Olivia Wiles, Isabela Albuquerque, Matthias Bauer, Su Wang, Jordi Pont-Tuset, and Aida Nematzadeh. Evaluating numerical reasoning in text-to-image models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2406.14774.

Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. GenEval 2: Addressing benchmark drift in text-to-image evaluation. *arXiv preprint arXiv:2512.16853*, 2025. URL https://arxiv.org/abs/2512.16853.

Kyuyoung Kim, Jongheon Jeong, Minyong An, Mohammad Ghavamzadeh, Krishnamurthy Dvijotham, Jinwoo Shin, and Kimin Lee. Confidence-aware reward optimization for fine-tuning text-to-image models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2404.01863.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.01569.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2411.15124. arXiv:2411.15124.

Sifan Li, Yujun Cai, Hongkai Chen, and Yiwei Wang. GeoSVG-RL: Geometry-Aware Reinforcement Learning for Layout-Constrained Text-to-SVG Diagram Generation. *arXiv preprint arXiv:2605.25447*, 2026. URL https://arxiv.org/abs/2605.25447.

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 (ECCV)*, 2024. URL https://arxiv.org/abs/2404.01291.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a. URL https://arxiv.org/abs/2505.05470.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Yunan Huang, Mozhi Zhang, Pengyu Zhao, Junjie Yan, and Junxian He. SynLogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. URL https://arxiv.org/abs/2505.19641. arXiv:2505.19641.

Yuhang Ma, Yunhao Shui, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. HPSv3: Towards wide-spectrum human preference score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. URL https://arxiv.org/abs/2508.03789.

OpenAI. GPT-Image-1 Mini. OpenAI API model documentation, 2025. URL https://developers.openai.com/api/docs/models/gpt-image-1-mini. OpenAI API model documentation. Model ID gpt-image-1-mini, released October 6, 2025 (OpenAI API changelog).

OpenAI. GPT-Image-2. OpenAI API model documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-image-2. OpenAI API model documentation. Model ID gpt-image-2, snapshot gpt-image-2-2026-04-21.

OpenAI. GPT Image 2.5 Sunburst. OpenAI API model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-image-2.5-sunburst. OpenAI API model documentation. Model ID gpt-image-2.5-sunburst, snapshot gpt-image-2.5-sunburst-2026-09-08; released September 8, 2026 together with gpt-image-2.5-flare.

OpenAI. ChatGPT Images 2.5 system card. OpenAI Deployment Safety Hub, 2026c. URL https://deploymentsafety.openai.com/chatgpt-images-2-5. OpenAI Deployment Safety Hub, published September 8, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2307.01952.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2507.02833. arXiv:2507.02833.

Qwen Team. Qwen-Image-2512: Finer details, greater realism. Qwen blog post, 2025. URL https://qwen.ai/blog?id=qwen-image-2512. Blog post, December 2025; model: https://huggingface.co/Qwen/Qwen-Image-2512.

Juan A. Rodriguez, Haotian Zhang, Abhay Puri, Aarash Feizi, Rishav Pramanik, Pascal Wichmann, Arnab Mondal, Mohammad Reza Samsami, Rabiul Awal, Perouz Taslakian, Spandana Gella, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Rendering-Aware Reinforcement Learning for Vector Graphics Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.20793.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.11487.

Michael Saxon, Fatima Jahara, Mahsa Khoshnoodi, Yujie Lu, Aditya Sharma, and William Yang Wang. Who evaluates the evaluations? objectively scoring text-to-image prompt coherence metrics with T2IScoreScore (TS2). In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2404.04251.

Christoph Schuhmann. LAION-aesthetics predictor (improved-aesthetic-predictor). GitHub repository, 2022. URL https://github.com/christophschuhmann/improved-aesthetic-predictor. See also https://laion.ai/blog/laion-aesthetics/.

Stability AI. Introducing Stable Diffusion 3.5. Stability AI blog post, 2024. URL https://stability.ai/news-updates/introducing-stable-diffusion-3-5. Blog post, October 22, 2024 (SD3.5 Large, Large Turbo, Medium).

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.24760. Spotlight. arXiv:2505.24760.

Tencent Hunyuan Team. HunyuanImage 2.1: An efficient diffusion model for high-resolution (2K) text-to-image generation. Tencent Hunyuan GitHub repository, 2025. URL https://github.com/Tencent-Hunyuan/HunyuanImage-2.1. Citation as given in the official repository; no technical report.

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 (CVPR)*, 2024. URL https://arxiv.org/abs/2311.12908.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation. *arXiv preprint arXiv:2503.05236*, 2025. URL https://arxiv.org/abs/2503.05236.

Olivia Wiles, Chuhan Zhang, Isabela Albuquerque, Ivana Kajić, Su Wang, Emanuele Bugliarello, Yasumasa Onoe, Pinelopi Papalampidi, Ira Ktena, Chris Knutsen, Cyrus Rashtchian, Anant Nawalgaria, Jordi Pont-Tuset, and Aida Nematzadeh. Revisiting text-to-image evaluation with Gecko: On metrics, prompts, and human ratings. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2404.16820.

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. URL https://arxiv.org/abs/2508.02324.

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. URL https://arxiv.org/abs/2306.09341.

Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. ConceptMix: A compositional image generation benchmark with controllable difficulty. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2408.14339.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution image synthesis with linear diffusion transformers. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.10629.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2304.05977.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025. URL https://arxiv.org/abs/2505.07818.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research (TMLR)*, 2022. URL https://arxiv.org/abs/2206.10789.

Ziyi Zhang, Sen Zhang, Yibing Zhan, Yong Luo, Yonggang Wen, and Dacheng Tao. Confronting reward overoptimization for diffusion models: A perspective of inductive and primacy biases. In *International Conference on Machine Learning (ICML)*, pp. 60396–60413, 2024. URL https://arxiv.org/abs/2402.08552.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023. URL https://arxiv.org/abs/2311.07911.

Sashuai Zhou, Qiang Zhou, Junpeng Ma, Yue Cao, Ruofan Hu, Ziang Zhang, Xiaoda Yang, Zhibin Wang, Jun Song, Cheng Yu, Bo Zheng, and Zhou Zhao. SpatialReward: Verifiable Spatial Reward Modeling for Fine-Grained Spatial Consistency in Text-to-Image Generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2603.22228.

A TASK REPRESENTATION

This section lists the constraint library of §2.1 and gives a complete example task.

A.1 CONSTRAINT LIBRARY

Table 7 lists all 46 constraint types and their contributions to structural complexity. Let $L(n) = 1 + \log_2 n$; n_i is the number of visible instances in object group i, N_l is the number of objects checked by layout constraint l, N_r is the total number of visible instances in the distinct groups referenced by relation r, and m_r is the number of required one-to-one matches.

Family	Exact constraint types	Contribution to $C(s)$
Grounding	`color_attribute`	$L(n_i)$ for the referenced group
	`shape_attribute`	$L(n_i)$ for the referenced group
	`color_shape_binding`	No additional term; the bound group's color and shape terms already account for it
Cardinality	`exact_count`	$L(n_i)$ for the referenced group
	`same_count`; `more_than_count`; `fewer_than_count`	$L(N_r)$
	`times_as_many`	$L(N_r) + \log_2 k$, $k \in \{2, 3, 4, 5\}$
Spatial	`absolute_region`; `grid_occupancy`	$L(N_l)$
	`left_of`; `right_of`; `above`; `below`; `all_left_of`; `all_right_of`; `all_above`; `all_below`; `leftmost`; `rightmost`; `topmost`; `bottommost`; `between`; `same_row`; `same_column`; `all_same_row`; `all_same_column`; `not_all_same_row`; `not_all_same_column`; `closer_than`; `farther_than`	$L(N_r)$
Size	`larger_than`; `smaller_than`; `same_size`; `all_larger_than`; `all_smaller_than`; `all_same_size`; `largest`; `smallest`	$L(N_r)$
	`not_all_same_size`	$2 \log_2 N_r$
Topology	`touching`; `not_touching`; `inside`; `contains`	$L(N_r)$
	`each_inside`; `each_contains`	m_r

Table 7: The exact 46 constraint types and their contributions to structural complexity. Types with the same cost rule share a row. The background and forbidden-content constraints in $\mathcal{B}$ and $\mathcal{F}$ apply to every task and do not contribute to $C(s)$.

A.2 EXAMPLE TASK

This example gives one task in the representation of §2.1 and its prompt; Appendix C.2 gives the verifier code for each of its constraints.

Task. The task contains two object groups and seven constraints:

$\mathcal{G} = \{$`g1`, `g2`$\}$,
$\mathcal{A} = \{$`color_attribute(g1; purple)`, `shape_attribute(g1; circle)`, `exact_count(g1; 1)`,
$= \{$`color_attribute(g2; yellow)`, `shape_attribute(g2; square)`, `exact_count(g2; 1)`,
`below(g1,g2)`$\}$.

The background constraints are $\mathcal{B} = \{$`background_color(pale pink)`$\}$, and the forbidden-content constraints are $\mathcal{F} = \{$`no_unrequested_objects`$\}$. For compactness, the implementation stores the color, shape, and count constraints of each group within the group's record, stores the background constraint as the background color, and applies `no_unrequested_objects` to every task, so its `forbidden` list holds only additional forbidden-content constraints:

```
{
  "background": {"color": "pale pink"},
  "objects": [
    {"id": "g1", "color": "purple", "shape": "circle", "count": 1},
    {"id": "g2", "color": "yellow", "shape": "square", "count": 1}
```

```
  ],
  "relations": [{"type": "below", "subject": "g1", "object": "g2"}],
  "forbidden": []
}
```

A group with `"count_mode": "relative"` has no `exact_count` constraint; its count is constrained only by relations such as `times_as_many`.

Prompt. The templates produce "Place the purple circle below the yellow square. Set the objects against a plain pale pink background; do not add other colored objects." Each constraint refers to groups by identifier, so the binding of each attribute to its object is unambiguous.

B TASK GENERATION

This section gives the complete generation procedure of §2.2, its validation steps, a worked example, and the construction of each dataset.

B.1 PROCEDURE

Algorithm 1 gives the procedure for one dataset. Its first steps implement steps 1–5 of §2.2, and the remaining steps are the validation checks of Appendix B.2.

Algorithm 1 Task generation and validation for one dataset.

initialize an empty task pool $\mathcal{Q}$
for each constructor index **do**
    sample background constraints $\mathcal{B}$, object groups $\mathcal{G}$ with colors, shapes, and counts, and forbidden-content constraints $\mathcal{F}$
    assign each object instance a center and a size to obtain the scene z
    $\mathcal{A}^* \leftarrow$ the instantiated constraints whose requirements hold in z
    for each active constraint set $\mathcal{A} \subseteq \mathcal{A}^*$ selected within the target complexity range **do**
        render the prompt p from $(\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F})$ and form $s = (\mathcal{G}, \mathcal{B}, \mathcal{A}, \mathcal{F}, p)$
        render the reference image $x^\star$ of z on the background specified by $\mathcal{B}$
        if p or the canonical form of s occurs in $\mathcal{Q}$ or in an excluded split **then**
            continue
        end if
        if $r_{\text{exact}}(x^\star, s) = 0$ **then**
            continue
        end if
        change one constraint of s to obtain a counterfactual task $\tilde{s}$
        if the changed constraint passes on $x^\star$ under $\tilde{s}$ **then**
            continue
        end if
        add $(s, x^\star)$ to $\mathcal{Q}$
    end for
end for
select tasks from $\mathcal{Q}$ to match the target distribution of the dataset (Appendix B.5)

Scene construction. The frozen constructors represent each object instance by its group identifier, integer center (u, v), and radius ρ on a 512×512 canvas. The default layout divides the canvas into six boxes arranged in three columns and two rows. For a group of n repeated objects, the constructor uses $\min(5, \lceil\sqrt{n}\rceil)$ columns and fills $\lceil n/\min(5, \lceil\sqrt{n}\rceil)\rceil$ rows at evenly spaced coordinates within its box. Relation-specific templates replace these default placements with fixed constructions for rows, columns, grids, contact, containment, order, proximity, extrema, and relative size. Seeded sampling selects counts, attributes, and template variants. The constructor then enumerates additional constraints that are true of the stored positions and sizes.

B.2 VALIDATION

Every retained task passes three checks.

Reference image. The generator renders the scene and requires the released verifier to accept it, which confirms that the pixel rendering preserves every constraint that holds in the scene.

Counterfactual. The generator changes one constraint while holding the reference image fixed and requires the verifier to reject the image under the changed task. For VVRBENCH, the changed constraint is evaluated on its own; for Challenge and the scene-first training candidates, the generator inverts one constraint and removes the other relation and layout constraints that could conflict with the inversion.

Deduplication. Deduplication uses normalized prompts and canonical tasks formed by renaming object identifiers in a fixed order, and it is applied jointly across each dataset and all excluded training and evaluation splits.

For each of the 10,000 VVRBENCH and 720 Challenge tasks, the verifier accepts the reference image and rejects the counterfactual.

B.3 WORKED EXAMPLE

This example traces one scene through the five steps of §2.2. It is the output of the scene-first generator for enumeration index 53 with the default seed; every value below is produced by the code.

Step 1: scene. The generator samples $\mathcal{B} = \{\texttt{background_color(white)}\}$ and three object groups with seven objects in total on a 512×512 canvas; $\mathcal{F} = \{\texttt{no_unrequested_objects}\}$.

Group	Color	Shape	Count	Centers (radius 9 px)
g0	cyan	circle	3	$(28, 34)$, $(148, 34)$, $(28, 214)$
g1	yellow	square	2	$(196, 34)$, $(316, 34)$
g2	pink	triangle	2	$(364, 34)$, $(484, 34)$

Steps 2 and 3: satisfiable constraint set. The generator instantiates each constraint type on the group tuples of its arity and keeps the instantiated constraints that hold in the scene. The resulting set $\mathcal{A}^*$ contains the nine unary color, shape, and count constraints of the three groups and the following fifteen constraints:

- count comparisons: `more_than_count(g0,g1)`, `more_than_count(g0,g2)`, `same_count(g1,g2)`;
- order: `all_left_of(g0,g1)`, `all_left_of(g0,g2)`, `all_left_of(g1,g2)`;
- alignment: `not_all_same_row(g0)`, `not_all_same_column(g0)`, `all_same_row(g1)`, `not_all_same_column(g1)`, `all_same_row(g2)`, `not_all_same_column(g2)`;
- regions: `absolute_region(g0;top)`, `absolute_region(g1;top)`, `absolute_region(g2;top)`.

For each pair of groups, the generator adds the one count comparison that holds and a direction only when the groups are separated by at least 16 px along that axis, a margin wider than the verifier's 12 px.

Step 4: active constraints. The generator adds at most four constraints from $\mathcal{A}^*$ to the unary ones, visiting constraint types in a fixed rotated order and skipping any constraint that would exceed the target complexity range. Every prefix of this sequence whose complexity lies in the target range is a task, so this scene yields four nested tasks. The largest adds `all_same_row(g2)`, `not_all_same_row(g0)`, `not_all_same_column(g2)`, and `same_count(g1,g2)`. Because `same_count` fixes the number of yellow squares relative to the pink triangles, `g1` loses its

`exact_count` constraint, so $\mathcal{A}$ contains twelve constraints: color and shape for all three groups, `exact_count(g0; 3)`, `exact_count(g2; 2)`, and the four added constraints. Its structural complexity is

$$C(s) = \underbrace{3L(3)}_{\texttt{g0}} + \underbrace{2L(2)}_{\texttt{g1}} + \underbrace{3L(2)}_{\texttt{g2}} + \underbrace{L(2) + L(3) + L(2) + L(4)}_{\text{added constraints}} = 27.34,$$

with $L(n) = 1 + \log_2 n$.

Step 5: prompt. The templates render the four nested tasks with different sentence frames:

- "The image should contain three cyan circles, two yellow squares, and two pink triangles. Arrange all the pink triangles in one row. Use a plain white background and no other colored objects."
- "Show three cyan circles, two yellow squares, and two pink triangles. Arrange all the pink triangles in one row. Arrange all the cyan circles so they are not all in the same row. Keep the background plain white, with no additional colored objects."
- "Create an image with three cyan circles, two yellow squares, and two pink triangles. Arrange all the pink triangles in one row. Arrange all the cyan circles so they are not all in the same row. Arrange all the pink triangles so they are not all in the same column. Set the objects against a plain white background; do not add other colored objects."
- "Draw three cyan circles and two pink triangles. Use the same number of yellow squares and pink triangles. Arrange all the pink triangles in one row. Arrange all the cyan circles so they are not all in the same row. Arrange all the pink triangles so they are not all in the same column. Use a plain white background and no other colored objects."

The last prompt states no count for the yellow squares, matching the removal of their `exact_count` constraint.

Validation. The generator renders the scene as the reference image in Figure 7, and the released verifier accepts it for all four tasks. The counterfactual of each task replaces `all_same_row(g2)` with `not_all_same_row(g2)` and removes the other added constraints; the verifier rejects the same image under every counterfactual.

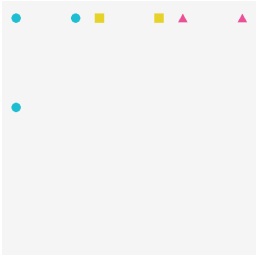

Figure 7: Reference image of the example scene.

B.4 Structural complexity

With $L(n) = 1 + \log_2 n$, the complexity of a task is

$$C(s) = \sum_{a \in \mathcal{A}} c(a; s), \tag{4}$$

where the cost $c(a; s)$ of each constraint type is given in Table 7. Color, shape, and exact count cost $L(n)$ for a group of n objects; most relations and layouts cost $L(N)$ for the N objects they compare; a count ratio by factor k adds $\log_2 k$; one-to-one containment costs the number of required matches; and within-group size variation costs $2 \log_2 N$. Every task has one background-color constraint and the forbidden-content constraint `no_unrequested_objects`, so the constraints in $\mathcal{B}$ and $\mathcal{F}$ are excluded.

Complexity as a predictor of failure. For each model we compute the AUC with which a single task feature separates unsolved from solved VVRBench tasks, and the McFadden R^2 of a logistic regression of exact success on that feature. The features are $C(s)$, the number of color, shape, relation, and layout constraints, and the numbers of object instances, object groups, and relations. The 19 models are the ten models of Table 3 other than Sana and SDXL, which solve fewer than three tasks, and the nine post-trained SD3.5-M models of Table 13. $C(s)$ has the highest AUC and R^2 for 18 models, with median AUC 0.857 and R^2 0.283; the number of color, shape, relation, and layout constraints follows with 0.829 and 0.232, and the number of object instances with 0.820 and 0.227.

B.5 DATASETS

Dataset parameters. No dataset generator uses model outputs or benchmark scores. Object groups are drawn from eight foreground colors, three shapes, counts from one through ten, and seven backgrounds, and reference images are constructed on a 512×512 canvas. The nine generation strata are quantity, binding, location, direction and order, between, proximity, size, structured layout, and topology. These strata govern dataset sampling; the verifier taxonomy remains the five-family hierarchy in Table 1.

VVRBENCH. The construction crosses the nine generation strata and their compositions with five generator settings. Each individual stratum receives 50 tasks per level, and each composition order of two, three, and four to six strata receives 600 tasks per level, divided evenly over stratum combinations. The selection rule then caps the number of tasks at each integer complexity by removing tasks from the most populated levels, and adds single object tasks at levels the generator does not otherwise reach.

Challenge. The first requirement of each candidate cycles through all 46 constraint types, and each reference image is constructed to satisfy it. The generator then adds at most one constraint of each type, up to six relation or layout requirements, and skips duplicate relations and combinations that the verifier treats as incompatible. Every prefix of the added requirements is a candidate, so several Challenge tasks can share one reference image. The selection program requires exactly 20 tasks at each integer complexity from 45 through 80, at least 20 occurrences of every nongrounding constraint type, and at most two thirds of total complexity from color, shape, and exact count requirements. Subject to these constraints, it balances structural complexity across families and constraint-type frequencies within each family. The selected tasks allocate 15.0% of structural complexity to cardinality, 35.1% to spatial constraints, 25.0% to size, and 24.9% to topology.

VVR-Easy. Each task adds one constraint type from one generation stratum to the color, shape, and count constraints of its object groups, so it exercises at most one constraint family beyond them; 71% of tasks contain one relation and the rest none. Every task has complexity at most 20. The nine strata receive equal quotas, and constraint types within each stratum receive fixed quotas.

VVR-Matched. VVR-Matched uses the generation strata of VVRBENCH. Tasks are allocated to strata in proportion to the benchmark, and family and constraint-type frequencies are equalized within each stratum. The corpus is accepted only if a Kolmogorov–Smirnov test finds its complexity distribution matched to the benchmark.

Metadata. Each task records the continuous complexity and its color, shape, count, layout, and relation components; active constraint families; atomic requirements; prompt and task hashes; the reference rendering hash; and the counterfactual task and score.

C VERIFIER

This section describes the verifier of §2.3: object extraction, the program verifiers, the scores and training reward, and its validation.

C.1 PIXEL-TO-OBJECT EXTRACTION

The verifier estimates the background as the median color along the image boundary and uses variation among those boundary pixels to set a background-relative foreground threshold. It converts the image to HSV and assigns sufficiently saturated foreground pixels to fixed, nonoverlapping hue ranges for the eight supported colors. Low-confidence and background-like pixels are excluded. On each binary color mask, erosion followed by dilation removes isolated foreground pixels, and dilation followed by erosion fills small holes and narrow breaks. The implementation scans the cleaned mask and uses flood fill from each unlabeled foreground pixel, traversing horizontal, vertical, and diagonal neighbors. Every maximal set reached by one traversal becomes a candidate object.

Each component is described by its area, centroid, bounding box, boundary, aspect ratio, bounding-box occupancy, convexity, convex-hull vertex count, number of holes, and offset between its centroid and bounding-box center. A fixed geometric classifier converts these measurements into circle, square, and triangle scores. Circle scores favor approximately equal width and height, high convexity, and rounded contours; square scores cover both filled axis-aligned boxes and centered, convex rotated squares; triangle scores use their characteristic bounding-box occupancy and off-center centroid. A single hole provides additional evidence for an outlined circle or square. Components that are too small, narrow, or weakly supported by the requested color and shape are removed. If several requested groups have the same color but different shapes, each component is assigned exclusively to the shape receiving its highest score.

The frozen rules include three safeguards for imperfect generations. First, background-adaptive contrast and calibrated hue boundaries handle shading and colors near category boundaries. Explicit boundary rules separate pale, low-saturation red from pink and muted blue-violet from bright blue. Second, morphological cleanup and a shape-conditioned fallback mask recover objects with fragmented or blurred color regions without allowing one component to satisfy two color groups. Third, robust extents use the 5th and 95th percentiles of component coordinates, reducing sensitivity to stray boundary pixels. The exact thresholds are fixed in the released verifier. Appendix C.4 reports calibration and held-out tests covering ambiguous colors, irregular contours, compression, blur, touching objects, and threshold-adjacent cases.

C.2 Program verifiers

The listings below are excerpts from the released `vvr_bench/verifier.py` for the four constraint types in the example task of Appendix A.2. Helper functions are named but not shown. The extraction step of §2.3 provides each group's matched objects as components with a centroid and a score for each shape; `_estimate_repeated_group_count` counts the objects of a group and counts a connected region whose area is close to an integer multiple of one object's area as that many touching objects.

For `exact_count` and `color_attribute`, the verifier compares the estimated count with the target and requires at least one object of the group's color:

```
count_pred = _estimate_repeated_group_count(count_components, shape)
if count_is_exact: # exact_count
   count_error, count_score = _score_exact_count(count_pred, target_count)
else: # count_mode == "relative": presence only
   count_error = 0.0 if count_pred >= 1 else 1.0
   count_score = 1.0 if count_pred >= 1 else 0.0

color_presence_score = min(1.0, float(count_pred)) # color_attribute
color_presence_strict = count_pred >= 1
if (len(spec.get("objects", [])) > 1 and not color_presence_strict
      and any(_component_identity_compatible(c, shape, image.shape[:2])
            for c in fallback_components)):
   color_presence_score = 1.0
   color_presence_strict = True

def _score_exact_count(observed_count, target_count):
   error = abs(observed_count - target_count)
   score = max(0.0, 1.0 - error / max(target_count, 1))
   return error, float(score)
```

The `exact_count` constraint passes when `count_error` is zero, and `color_attribute` passes when `color_presence_strict` holds.

For `shape_attribute`, the verifier averages the requested shape's score over the group's objects and compares it with a shape-specific threshold:

```
shape_score = _score_shape_attribute(shape, selected, allow_occluded_triangle=True)
shape_strict = shape_score >= _shape_presence_threshold(shape)

def _score_shape_attribute(shape, components, *, allow_occluded_triangle=False):
   if not components:
      return 0.0
   return float(np.mean([
      _effective_shape_score(component, shape) if allow_occluded_triangle
      else component.shape_scores.get(shape, 0.0)
```

```
      for component in components
   ]))

def _shape_presence_threshold(shape):
   return 0.20 if shape == "triangle" else 0.40
```

For `below`, the verifier first rejects nested referents, where one group's object lies inside the other's, and then compares the mean vertical centroids with a margin; image coordinates increase downward:

```
nested = any(
   _bbox_intersection_fraction(first, second) >= 0.98
   and math.dist(first.centroid, second.centroid)
      <= 0.80 * min(_component_extent(first), _component_extent(second))
   and max(first.shape_scores.values(), default=0.0) >= 0.40
   and max(second.shape_scores.values(), default=0.0) >= 0.40
   for first in subject
   for second in obj
)
if nested:
   return 0.0, {**result, "strict_pass": False, "nested_referents": True}

subj_y = float(np.mean([comp.centroid[1] for comp in subject]))
obj_y = float(np.mean([comp.centroid[1] for comp in obj]))
margin = float(relation.get("margin_px", 24.0))
delta = subj_y - obj_y
score = min(1.0, max(0.0, delta / max(margin, 1.0)))
strict_pass = delta >= margin
```

The verifier also reports a color–shape binding score for each group, computed from its color and shape scores; binding adds no structural complexity (Table 7). The image passes this task when all seven constraints in $\mathcal{A}$ and the constraints in $\mathcal{B}$ and $\mathcal{F}$ pass (§2.3).

Constraint measurements. The verifier assigns a fixed geometric meaning to each relational phrase in the prompt templates. Let h be the shorter image side and e the largest visible extent among the objects compared.

- *Same row* (column): the vertical (horizontal) spread of the object centroids is at most $\max(0.04h,\, 0.55e)$.
- *Between*: let t be the position of the subject's centroid projected onto the segment joining the two reference centroids, and d its distance from that segment. The relation holds when $\max(0, 1 - d/0.25\ell)\cdot\mathbf{1}[0.15 \le t \le 0.85] \ge 0.70$, where ℓ is the segment length and the indicator is replaced by a linear decay outside the interval for partial credit.
- *Closer than*: distance is the minimum Euclidean distance between component boundaries, which reflects the visible gap between objects of different sizes. The nearer distance must be at most 0.90 of the farther distance and at least 4 pixels smaller.
- *Largest* (smallest) *colored object*: the subject's visual extent, defined below, is compared with that of every visible colored component in the image, including components that match no requested group.

Relative size uses visual extent, the geometric mean of a component's width and height measured between the 5th and 95th percentiles of its pixel coordinates. The benchmark compares sizes only relative to other objects, because calibration found no stable human decision boundary for absolute size.

Objects and unmatched components. Requested objects are matched by color and shape to connected visual components. Count compares the number of matched components with the requested cardinality. Any remaining visible colored component is unmatched, so an extra copy of a requested object lowers both the count score and the unmatched-component score.

C.3 Scores and training reward

The dense reward r_{dense} of Eq. 3 combines the partial-credit scores q_a in two levels. Each requested object group i receives

$$r_i = 0.45r_{\text{count},i} + 0.20r_{\text{shape},i} + 0.20r_{\text{layout},i} + 0.15r_{\text{size},i}, \tag{5}$$

where each term is the partial-credit score of that group's constraints of the given kind, and r_{obj} is the mean across groups. Let r_{rel} be the mean partial-credit score of the relations, I_{rel} indicate whether the task has a relation, and r_{forbid}, r_{extra}, and r_{bg} be the scores of the forbidden-content, unmatched-component, and background constraints. The weighted sum in Eq. 3 is

$$\sum_a w_a q_a = \frac{0.55 r_{\mathrm{obj}} + 0.15 I_{\mathrm{rel}} r_{\mathrm{rel}} + 0.15 r_{\mathrm{forbid}} + 0.10 r_{\mathrm{extra}} + 0.05 r_{\mathrm{bg}}}{0.85 + 0.15 I_{\mathrm{rel}}}, \tag{6}$$

and the penalty factor is

$$\psi = g_{\mathrm{count}} g_{\mathrm{rel}} g_{\mathrm{extra}} g_{\mathrm{forbid}}, \tag{7}$$

with

$$\begin{aligned} g_{\mathrm{count}} &= 0.10 + 0.90 r_{\mathrm{count}}, \\ g_{\mathrm{rel}} &= \begin{cases} 1, & I_{\mathrm{rel}} = 0, \\ 0.25 + 0.75 r_{\mathrm{rel}}, & I_{\mathrm{rel}} = 1, \end{cases} \\ g_{\mathrm{extra}} &= \max\left(0, 1 - \frac{n_{\mathrm{extra}}}{\max(N_{\mathrm{target}}, 1)}\right), \\ g_{\mathrm{forbid}} &= r_{\mathrm{forbid}}. \end{aligned} \tag{8}$$

Here r_{count} is the mean group-level count score, n_{extra} is the number of unmatched components, and N_{target} is the requested object count.

C.4 Verifier validation

The verifier passes 290 historical edge cases, 360 direct checks, 14,788 metamorphic checks, a 320-case matrix of single constraints, 200 constructed cases at decision thresholds, and geometry tests for repeated-group size, containment, contact, and relation inverses. These tests cover blur, compression, low contrast, irregular contours, touching and merged components, missing objects, and reversed relations. During development, 3,947 human decisions set the decision boundary of each perceptual predicate: the hue range of every color name, the margin at which two objects touch, the contour tolerances that separate circles, squares, and triangles, and the ratio at which one object counts as larger than another.

Two human audits test the verifier on generated images. Each audit image tests one constraint, labeled by one annotator without seeing the verifier's decision. The larger audit contains 512 images of 128 prompts generated by pretrained SD3.5-M, FLUX.1-dev, and two SD3.5-M models trained with earlier VVR rewards. The verifier version frozen before this audit agrees with 454 of 508 decisive labels (89.4%, Cohen's $\kappa = 0.78$), and an earlier audit of 528 images agrees on 421 of 452 (93.1%, $\kappa = 0.86$). After calibration that used the larger audit, the released verifier, which scores every result in this paper, agrees with 481 of its 508 labels (94.7%, $\kappa = 0.89$).

D Benchmark Evaluation Details

D.1 Evaluation details

Models generate at their native resolution. We score VVRBench images at 512×512 and Challenge images at 1024×1024, which preserves boundaries and small objects in dense scenes. API models receive one request per prompt; transient errors are retried, and completed responses are never resampled. Appendix D.3 reports how often each API model returned no image.

Generation settings. Table 8 lists the settings of every model. Open-weight models generate at 1024×1024, except HunyuanImage-2.1 at 2048×2048, and all post-trained SD3.5-M models use the SD3.5-M settings. The seed of each prompt is the first 32 bits of the SHA-256 hash of a fixed base seed and the prompt identifier, so every open-weight model receives the same seed for the same prompt. The OpenAI image API has no temperature parameter, and Gemini models are called with a 1:1 aspect ratio and default values for temperature and all other sampling parameters.

Table 8: Generation settings. Guidance is the classifier-free guidance scale.

Open-weight model	Steps	Guidance	API model	Settings
FLUX.2-dev	50	4.0	GPT-Image-2.5-Sunburst (2026-09-08)	medium quality, 1024 × 1024
HunyuanImage-2.1	50	3.5	GPT-Image-2 (2026-04-21)	medium quality, 1024 × 1024
Qwen-Image-2512	50	4.0	GPT-Image-1-mini	medium quality, 1024 × 1024
HiDream-I1-Full	50	5.0	Gemini-3-Pro-Image	1K
FLUX.1-dev	28	3.5	Gemini-3.1-Flash-Image	1K
FLUX.1-schnell	4	0.0	Gemini-3.1-Flash-Lite-Image	1K
SD3.5 Medium	40	4.5	Gemini-2.5-Flash-Image	model default (1K)
SD3.5 Large	40	4.5		
SDXL 1.0	40	5.0		
Sana 1.6B	20	4.5		

D.2 Complete VVRBench-Fast results

Table 9 reports the complete complexity breakdown underlying Figure 2.

Table 9: Accuracy (%) of API models on VVRBench-Fast, an 820 task subset of VVRBench with 20 tasks at each attainable integer complexity from 3 to 44. The GPT-Image-2 models exceed 80% overall but fall to 51% to 57% in C_5. Gemini models reach 37% to 47%. Subscripts are 95% confidence margins.

Model	Accuracy (%) ↑	3 to 10	11 to 18	19 to 26	27 to 35	36 to 44
GPT-Image-2.5-Sunburst	**84.51**$_{\pm 2.64}$	**100.00**$_{\pm 2.67}$	**98.75**$_{\pm 3.19}$	**96.25**$_{\pm 4.19}$	**77.22**$_{\pm 6.66}$	**56.67**$_{\pm 7.30}$
GPT-Image-2	82.20$_{\pm 2.77}$	98.57$_{\pm 3.63}$	**98.75**$_{\pm 3.19}$	95.62$_{\pm 4.38}$	74.44$_{\pm 6.84}$	50.56$_{\pm 7.24}$
Gemini-3.1-Flash-Image	47.20$_{\pm 3.42}$	85.71$_{\pm 6.75}$	60.00$_{\pm 7.74}$	51.25$_{\pm 7.68}$	30.56$_{\pm 7.08}$	18.89$_{\pm 6.35}$
Gemini-2.5-Flash-Image	45.61$_{\pm 3.42}$	76.43$_{\pm 7.68}$	65.62$_{\pm 7.65}$	51.25$_{\pm 7.68}$	31.11$_{\pm 7.10}$	13.33$_{\pm 5.74}$
Gemini-3.1-Flash-Lite-Image	42.07$_{\pm 3.41}$	85.71$_{\pm 6.75}$	56.25$_{\pm 7.74}$	40.00$_{\pm 7.74}$	25.00$_{\pm 6.80}$	14.44$_{\pm 5.88}$
Gemini-3-Pro-Image	37.20$_{\pm 3.36}$	62.14$_{\pm 8.26}$	52.50$_{\pm 7.71}$	38.75$_{\pm 7.73}$	26.67$_{\pm 6.90}$	13.33$_{\pm 5.74}$
GPT-Image-1-mini	32.44$_{\pm 3.28}$	92.86$_{\pm 5.51}$	60.00$_{\pm 7.74}$	16.88$_{\pm 6.56}$	4.44$_{\pm 4.08}$	2.78$_{\pm 3.56}$

D.3 Responses without an image

Some API models return text instead of an image, typically stating that the prompt is contradictory or too complex. These abstentions are incorrect: every task is satisfiable, because its reference image passes the verifier. On Challenge, Gemini-2.5-Flash-Image returned no image for 39 of 720 prompts, Gemini-3.1-Flash-Image for 3, Gemini-3.1-Flash-Lite-Image for 2, and Gemini-3-Pro-Image for 1; on VVRBench-Fast, Gemini-2.5-Flash-Image did so for 14 of 820 prompts and Gemini-3-Pro-Image for 3. The GPT models always returned an image. Each such response scores zero, and the result files keep its text. Of the 62 responses without an image, 48 contain text and 14 are empty. Three examples follow, with the instructions and responses verbatim.

Gemini-2.5-Flash-Image on VVRBench-Challenge

Instruction. "Draw nine red squares and three blue squares. Use more orange squares than red squares. Arrange all the red squares so they are not all in the same row. Arrange all the red squares in one column. Arrange all the blue squares so they are not all in the same row. Arrange all the blue squares so they are not all in the same column. Make every red square the same size as every blue square. Set the objects against a plain beige background; do not add other colored objects."

Model response. "I cannot fulfill your request. You have provided contradictory instructions regarding the arrangement of the red squares. You asked that 'all the red squares so they are not all in the same row' and also that 'all the red squares in one column.' These two statements cannot both be true simultaneously."

Why it is wrong. The two requirements are compatible: nine red squares stacked in one column share a column and, because they are stacked vertically, are not all in the same row.

Gemini-3-Pro-Image on VVRBENCH-Challenge

Instruction. "Show nine orange squares and two pink triangles. Use more cyan squares than orange squares. Arrange all the pink triangles in one row. Arrange all the cyan squares so they are not all in the same row. Arrange all the cyan squares so they are not all in the same column. Make every cyan square the same size as every orange square. Arrange all the cyan squares in row 1, column 1, row 1, column 2, row 1, column 3, row 1, column 4, row 2, column 2, row 2, column 3, row 2, column 4, row 3, column 1, row 3, column 2, and row 3, column 3 of a 3-by-4 grid. Set the objects against a plain pale cyan background; do not add other colored objects."

Model response. "I cannot generate the image because the constraints are contradictory. The prompt asks for a total of 19 squares (nine orange and ten cyan, as their specific positions are listed) to be arranged within a 3-by-4 grid, which has only 12 available spots." (Markdown emphasis removed.)

Why it is wrong. Only the ten cyan squares are assigned to grid cells, and ten cells are listed. The nine orange squares have no grid position and can be placed anywhere outside the grid.

Gemini-2.5-Flash-Image on VVRBENCH-Challenge

Instruction. "Show one pink square, one green circle, nine cyan circles, two yellow circles, and one purple square. Use four times as many cyan squares as yellow circles. Make the pink square contain the green circle. Place all the yellow circles above all the cyan squares. Place the purple square to the left of all the cyan squares. Place all the yellow circles to the right of the purple square. Place all the cyan squares in the right area. Keep the background plain black, with no additional colored objects."

Model response. "I'm sorry, I cannot fulfill that request. I am unable to keep track of that many rules for generating an image."

Why it is wrong. The task is long but satisfiable: its reference image passes the verifier, and Gemini-3.1-Flash-Image generated an image that satisfies every constraint of the same task.

D.4 PER-CONSTRAINT PASS RATES OF API MODELS

We score all 5,040 outputs of the seven API models on VVRBENCH-Challenge and record the pass-or-fail decision of every constraint check. The pass rate of a constraint type pools all of its checks, and responses without an image count as failures. Table 10 gives the pass rate of every constraint type.

D.5 COMPLEXITY-MATCHED FAMILY ANALYSIS

Table 11 tests whether a model loses accuracy on tasks that contain a constraint family, beyond what the tasks' complexity explains. For each model and family, it reports VVRBENCH accuracy on the tasks that contain the family and, in parentheses, the difference from tasks without the family at matched complexity. The largest negative differences identify family-specific weaknesses: GPT-Image-1-mini loses 9.7 points on tasks with **Spatial** constraints, HunyuanImage-2.1 loses 10.7 points with **Size** constraints, and FLUX.2-dev loses 4.0 points with **Cardinality** constraints. Models that solve few tasks show differences near zero.

To match complexity, we stratify VVRBENCH prompts by floored integer complexity and, within every stratum that contains tasks with and without family f, weight the accuracy of tasks without f by the number of tasks with f. The difference for model m is

$$\Delta_{m,f} = \sum_{c} w_{f,c} \left[\mathrm{Acc}_m(f, c) - \mathrm{Acc}_m(\neg f, c) \right], \tag{9}$$

where $w_{f,c}$ is the family present complexity distribution. The matched supports are 3,077 Grounding prompts (96% coverage), 3,297 Cardinality (98%), 6,713 Spatial (79%), 3,182 Size (100%), and 3,027 Topology (100%). Here "present" means that the benchmark sampled an explicit constraint from that family; ordinary object realization still appears throughout the benchmark. The background and forbidden-content constraints apply to every task, so they have no tasks without them to compare against.

Table 10: Pass rates (%) of API models for every constraint type on VVRBENCH-Challenge. n is the number of checks of each type per model, and Avg is the unweighted mean over the seven models. Responses without an image count as failures. Within each family, types are sorted by Avg. Cell shading is proportional to the pass rate.

Constraint type	n	GPT-Image-2.5-Sunburst	GPT-Image-2	Gemini-3.1-Flash-Lite	Gemini-3-Pro	Gemini-3.1-Flash	Gemini-2.5-Flash	GPT-Image-1-mini	Avg
Grounding									
color_shape_binding	3794	99	98	91	89	92	84	92	**92**
shape_attribute	3794	99	98	94	94	94	88	94	**94**
color_attribute	3794	99	99	98	99	98	94	95	**98**
Cardinality									
same_count	266	56	43	32	27	28	17	16	**31**
times_as_many	136	58	44	39	35	34	16	7	**33**
fewer_than_count	20	55	55	45	55	40	65	30	**49**
more_than_count	20	60	70	50	65	70	40	40	**56**
exact_count	3794	81	75	72	64	62	49	49	**64**
Spatial									
grid_occupancy	20	10	10	20	10	0	0	0	**7**
rightmost	20	40	30	55	50	60	20	30	**41**
leftmost	20	15	55	55	75	55	30	40	**46**
bottommost	20	75	75	55	40	55	40	40	**54**
between	20	85	60	45	55	60	50	30	**55**
topmost	20	80	85	55	65	50	45	45	**61**
all_same_column	20	100	95	80	50	60	50	60	**71**
all_right_of	122	86	78	78	82	74	69	55	**74**
all_left_of	281	88	79	78	83	75	72	57	**76**
closer_than	40	92	98	85	88	70	68	57	**80**
farther_than	20	95	90	85	80	80	60	80	**81**
right_of	20	100	95	90	85	85	70	70	**85**
all_same_row	32	97	94	84	78	94	88	78	**88**
absolute_region	384	98	97	91	92	90	77	80	**89**
all_below	57	98	96	89	91	88	82	81	**89**
all_above	102	100	99	91	93	91	82	82	**91**
same_column	20	100	95	95	95	100	90	65	**91**
not_all_same_row	136	98	96	96	90	94	85	91	**93**
same_row	20	100	100	100	95	90	90	80	**94**
not_all_same_column	145	99	98	97	91	95	85	90	**94**
below	20	95	95	100	95	100	75	100	**94**
left_of	20	100	100	95	95	85	95	95	**95**
above	20	100	100	95	100	100	75	95	**95**
Size									
smallest	20	75	80	5	10	15	20	5	**30**
all_same_size	487	74	50	50	44	48	31	32	**47**
same_size	88	98	91	89	85	80	68	70	**83**
all_larger_than	264	99	96	88	83	80	70	92	**87**
all_smaller_than	20	100	100	95	75	85	80	80	**88**
smaller_than	20	100	100	85	90	95	75	90	**91**
largest	20	100	100	85	100	90	75	90	**91**
larger_than	33	97	97	94	94	94	70	97	**92**
not_all_same_size	20	100	100	95	100	90	80	80	**92**
Topology									
each_contains	581	27	19	61	53	44	18	17	**34**
touching	20	50	70	60	50	70	20	0	**46**
each_inside	138	92	83	62	60	54	38	47	**62**
contains	20	100	100	95	95	90	60	100	**91**
inside	20	95	100	100	100	85	70	95	**92**
not_touching	259	100	100	99	100	99	89	97	**98**
Background and forbidden content									
no_unrequested_objects	720	62	51	56	40	35	30	46	**46**
background_color	720	100	100	96	94	93	93	99	**96**

Family tags can co-occur. As a sensitivity check, a linear probability model with all five family indicators and integer complexity fixed effects preserves the largest negative profiles. GPT-Image-1-mini Spatial changes from -9.7 to -14.6 points, FLUX.2-dev Cardinality from -4.0 to -5.9, and HunyuanImage-2.1 Size from -10.7 to -12.6. Positive associations for GPT-Image-2 are less stable under this adjustment.

Table 11: VVRBENCH accuracy (%) on prompts that contain each constraint family. In parentheses is the difference from prompts without that family at matched complexity. Models show distinct weaknesses: Spatial for GPT-Image-1-mini (−9.7), Size for HunyuanImage 2.1 (−10.7), and Cardinality for FLUX.2 dev (−4.0).

Model	Overall	Grounding	Cardinality	Spatial	Size	Topology
GPT-Image-2	86.9	86.8 (+0.5)	80.0 (+8.0)	85.4 (−4.8)	87.6 (+10.0)	85.6 (+7.7)
GPT-Image-1-mini	26.4	26.6 (+0.5)	14.2 (+1.8)	20.7 (−9.7)	19.2 (+2.5)	10.7 (−3.3)
FLUX.2-dev	19.1	19.0 (−0.5)	8.5 (−4.0)	16.3 (+0.3)	11.7 (−1.4)	9.1 (−1.2)
HunyuanImage-2.1	18.8	19.0 (+0.6)	10.3 (+0.0)	16.5 (−0.4)	6.5 (−10.7)	11.5 (−0.7)
Qwen-Image-2512	5.8	6.7 (+1.0)	2.2 (−0.7)	4.4 (+0.2)	2.4 (−1.3)	3.3 (+1.1)
HiDream-I1-Full	4.2	5.3 (+0.1)	2.0 (+0.1)	2.6 (−1.8)	1.6 (−0.5)	1.5 (+0.0)
FLUX.1-dev	3.9	5.2 (+0.4)	1.8 (+0.2)	2.6 (−0.0)	1.1 (−0.9)	1.8 (+0.6)
FLUX.1-schnell	2.9	3.8 (−0.1)	1.4 (+0.3)	1.8 (−0.1)	0.4 (−1.0)	1.0 (+0.3)
SD3.5 Medium	2.8	4.1 (−0.0)	1.3 (+0.0)	1.5 (−0.9)	0.9 (−0.1)	0.9 (+0.1)
SD3.5 Large	2.5	3.5 (+0.3)	1.7 (+0.4)	1.3 (−0.7)	0.5 (−0.5)	1.0 (+0.4)
SDXL 1.0	0.0	0.1 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	0.0 (+0.0)
Sana 1.6B	0.0	0.0 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	0.0 (+0.0)

D.6 FAILURE EXAMPLES

Figure 8 shows three failed VVRBENCH-Fast outputs of API models that add content the prompt excludes: vases, a bowl, and an apple; a lemon and a mug; and additional squares and shapes around the grid.



(a) (b) (c)

Figure 8: Failed API model outputs on VVRBENCH-Fast. Prompts: (a) “Place an orange circle in the top area. Set the objects against a plain pale pink background; do not add other colored objects.” (b) “Place a yellow square in the top area. Set the objects against a plain pale pink background; do not add other colored objects.” (c) “Arrange three purple squares in these cells of a 3-by-3 grid: top center, top right, and middle right. Set the objects against a plain black background; do not add other colored objects.”

E TRAINING SETUP

Table 12 reports the settings that define the optimization and reward distribution. The released resolved configurations and data manifests retain the remaining implementation metadata.

Table 12: Reproducible configuration for the final SD3.5 Medium post-training experiments. Reward proportions are fractions of prompt groups in each update; every rollout is scored only by the reward attached to its prompt.

Setting	Value
Trainable parameters	LoRA on the eight attention projections `add_k`, `add_q`, `add_v`, `add_out`, `k`, `q`, `v`, and `out`; rank 32 and $\alpha = 64$.
Generation	512×512 pixels; 25 denoising steps; classifier-free guidance 4.5; Gaussian sampling noise with level 0.7.
Rollout batch	32 prompt groups per update, 24 rollouts per prompt, and 768 generated images per update.
Flow GRPO	One inner epoch; advantages centered within each prompt group and divided by the standard deviation over the complete rollout batch; advantages clipped to $[-5, 5]$; policy-ratio clip 10^{-4}; KL coefficient 0.04. The first 24 of the 25 sampled transitions contribute to the update.
Optimization	AdamW; learning rate 3×10^{-4}; $\beta_1 = 0.9$, $\beta_2 = 0.999$; $\epsilon = 10^{-8}$; weight decay 10^{-4}; maximum gradient norm 1.0; FP16 mixed precision with TF32 enabled; exponential moving average.
Training duration	3,000 optimizer updates, corresponding to 96,000 prompt groups and 2,304,000 generated images.
VVR objective	Weights $(0.55, 0.15, 0.15, 0.10, 0.05)$ for object fidelity, relations, forbidden content, unmatched components, and background, respectively, multiplied by the penalty factor ψ in Eq. 7.
VVR complexity and data	Complexity follows Eq. 4. VVR-Easy contains 100,000 unique tasks with $C \leq 20$, each drawn from one generation stratum and containing at most one relation. VVR-Matched contains 100,000 unique tasks matched to the benchmark distribution over constraint family, family count, and complexity.

Training condition	Prompt-group and reward allocation	VVR tasks consumed
VVR-Easy	100% VVR-Easy	96,000
VVR-Matched	100% VVR-Matched	96,000
GenEval2	100% GenEval2	0
GenEval2 + VVR-Easy	50% GenEval2, 50% VVR-Easy	48,000
GenEval2 + VVR-Matched	50% GenEval2, 50% VVR-Matched	48,000
OCR	100% OCR	0
OCR + VVR-Easy	50% OCR, 50% VVR-Easy	48,000
Five-reward	20% each: GenEval, GenEval2, PickScore, OCR, UnifiedReward	0
Five-reward + VVR-Easy	$1/6$ each: the five rewards at left and VVR-Easy	16,000

E.1 Post-training evaluation

Each model generates one image per prompt with a fixed seed, except on GenEval, which uses four images per prompt.

Training-objective benchmarks. Each reward objective is evaluated on its own held-out benchmark. VVRBench accuracy uses the 10,000 VVRBench tasks. GenEval (Ghosh et al., 2023) uses its 553 prompts with four images each, for 2,212 images. GenEval2 (Kamath et al., 2025) uses its fixed 80-prompt held-out split. OCR (Liu et al., 2025a) uses 1,018 held-out text-rendering prompts scored by normalized edit accuracy. Each benchmark is reported in its own units.

Preference benchmarks. PickScore (Kirstain et al., 2023) is evaluated on the 500 unique prompts of the Pick-a-Pic v1 `validation_unique` split. HPSv2.1 (Wu et al., 2023) is evaluated on the complete HPDv2 benchmark, 800 prompts in each of four domains (anime, concept art, paintings, and photo), and reported as the unweighted mean of the four domain means.

Cross-domain panel. The remaining metrics use a shared panel of four prompt sets: all 200 DrawBench (Saharia et al., 2022) prompts and fixed 1,000-prompt subsets of PartiPrompts (Yu et al., 2022), DPG-Bench (Hu et al., 2024), and T2I-CompBench (Huang et al., 2023). On this panel we report HPSv3 (Ma et al., 2025), which has no canonical prompt benchmark, CLIPScore (Hessel et al., 2021), LAION aesthetic score (Schuhmann, 2022), ImageReward (Xu et al., 2023), and UnifiedReward (Wang et al., 2025). Each metric is averaged within a prompt set and then across the four sets, so the larger sets do not dominate. Only the five-reward objective trains on one of these metrics (UnifiedReward); together they test transfer to prompt distributions outside the training tasks.

F Complete RLVVR Results

F.1 VVRBench results by complexity

Table 13 gives the VVRBench accuracy of every trained model by complexity range; Figure 4 plots a subset.

Table 13: VVRBench accuracy (%) of SD3.5 M after post-training. Training on VVR raises accuracy from 2.81% to 28.27% with Easy tasks and 46.60% with Matched tasks. Matched training gives the largest gains at high complexity. Adding VVR-Easy to GenEval2, OCR, or the five-reward objective raises accuracy by a factor of three to seven. Subscripts are 95% confidence margins.

Training reward	Accuracy (%) ↑	C_1	C_2	C_3	C_4	C_5
SD3.5-M (pretrained)	$2.81_{\pm 0.34}$	$12.01_{\pm 1.47}$	$1.19_{\pm 0.59}$	$0.35_{\pm 0.37}$	$0.05_{\pm 0.23}$	$0.00_{\pm 0.19}$
GenEval2	$3.87_{\pm 0.40}$	$15.42_{\pm 1.61}$	$2.39_{\pm 0.78}$	$0.91_{\pm 0.52}$	$0.05_{\pm 0.23}$	$0.05_{\pm 0.23}$
GenEval2 + VVR-Easy	$21.82_{\pm 0.82}$	$54.51_{\pm 2.15}$	$32.05_{\pm 2.12}$	$14.08_{\pm 1.60}$	$6.41_{\pm 1.16}$	$1.10_{\pm 0.56}$
VVR-Easy	$28.27_{\pm 0.89}$	$\mathbf{67.72}_{\pm 2.04}$	$45.45_{\pm 2.23}$	$17.51_{\pm 1.73}$	$8.36_{\pm 1.29}$	$1.35_{\pm 0.60}$
VVR-Matched	$\mathbf{46.60}_{\pm 0.98}$	$67.68_{\pm 2.04}$	$\mathbf{59.17}_{\pm 2.21}$	$\mathbf{45.62}_{\pm 2.20}$	$\mathbf{38.39}_{\pm 2.15}$	$\mathbf{21.82}_{\pm 1.86}$
GenEval2 + VVR-Matched	$33.50_{\pm 0.93}$	$57.93_{\pm 2.13}$	$47.27_{\pm 2.23}$	$32.34_{\pm 2.09}$	$20.32_{\pm 1.82}$	$9.22_{\pm 1.34}$
OCR	$3.58_{\pm 0.38}$	$15.37_{\pm 1.61}$	$1.51_{\pm 0.65}$	$0.40_{\pm 0.39}$	$0.05_{\pm 0.23}$	$0.00_{\pm 0.19}$
OCR + VVR-Easy	$24.74_{\pm 0.86}$	$57.97_{\pm 2.13}$	$36.73_{\pm 2.18}$	$18.41_{\pm 1.76}$	$7.76_{\pm 1.26}$	$1.94_{\pm 0.70}$
Five-reward	$4.83_{\pm 0.44}$	$19.16_{\pm 1.75}$	$3.12_{\pm 0.87}$	$0.91_{\pm 0.52}$	$0.30_{\pm 0.35}$	$0.00_{\pm 0.19}$
Five-reward + VVR-Easy	$15.81_{\pm 0.73}$	$45.05_{\pm 2.14}$	$21.71_{\pm 1.90}$	$7.65_{\pm 1.25}$	$2.95_{\pm 0.84}$	$0.70_{\pm 0.47}$

F.2 Partial and joint constraint satisfaction

For every VVRBench task, we compute the mean partial-credit score of its count constraints and of its relations. A count score is one minus the relative count error, averaged over groups, and a relation score is the mean graded score of the task's relations. From these we report a partial score, the mean graded score, and the fraction of tasks in which every count or every relation is satisfied (Table 14). Relation columns use only the tasks with at least one relation. Figure 5 expresses VVR-Easy's values as the share of the gap between the pretrained model and VVR-Matched that VVR-Easy closes. In every range from C_3 to C_5, VVR-Easy closes more of the gap in partial scores than in the fraction of tasks with every constraint of a kind satisfied, and the difference grows with complexity.

Table 14: Partial and joint constraint satisfaction on VVRBENCH by complexity range. Partial scores are the mean graded count and relation scores; "all" is the fraction of tasks in which every count or every relation is satisfied. Relation columns use only tasks with at least one relation.

		Counts		Relations		
Range	Model	Partial	All	Partial	All	Accuracy (%)
C_1	Pretrained	0.76	0.46	0.29	0.19	12.01
	VVR-Easy	0.98	0.92	0.80	0.58	67.72
	VVR-Matched	0.98	0.93	0.80	0.57	67.68
C_2	Pretrained	0.70	0.17	0.25	0.13	1.19
	VVR-Easy	0.96	0.79	0.75	0.51	45.45
	VVR-Matched	0.97	0.84	0.80	0.53	59.17
C_3	Pretrained	0.69	0.10	0.20	0.06	0.35
	VVR-Easy	0.94	0.66	0.59	0.27	17.51
	VVR-Matched	0.98	0.88	0.77	0.41	45.62
C_4	Pretrained	0.66	0.06	0.21	0.03	0.05
	VVR-Easy	0.91	0.44	0.61	0.22	8.36
	VVR-Matched	0.98	0.76	0.82	0.42	38.39
C_5	Pretrained	0.61	0.02	0.21	0.02	0.00
	VVR-Easy	0.87	0.15	0.57	0.13	1.35
	VVR-Matched	0.97	0.48	0.85	0.41	21.82

F.3 EXTERNAL TASK, QUALITY, AND ALIGNMENT METRICS

Table 15 extends Table 5 to all nine trained models.

Table 15: Complete final evaluation of the pretrained model and nine post-training conditions. PickScore uses the Pick-a-Pic v1 validation benchmark and HPSv2.1 uses the official four-domain HPDv2 benchmark. HPSv3, CLIPScore, aesthetic score, ImageReward, and UnifiedReward are macro-averaged across the shared cross-domain panel. VVR accuracy, GenEval, GenEval2, and OCR retain their native units. Every HPSv3 value, including the pretrained one, is the mean over the four cross-domain prompt sets. Images of the pretrained model use sampling seed 42; images of trained models use seed 20260912, except for PickScore and HPSv2.1, which use seed 42 for every model.

Training reward	VVR	GenEval	GenEval2	OCR	PickScore	HPSv2.1	HPSv3	CLIPScore	Aesthetic	ImageReward	UnifiedReward
Pretrained	0.028	0.616	0.237	0.476	0.841	0.300	7.689	0.956	5.517	0.929	0.636
GenEval2	0.039	0.688	0.454	0.501	0.848	0.297	8.289	0.974	5.523	1.110	0.635
GenEval2 + VVR-Easy	0.218	0.718	0.478	0.532	0.848	0.302	8.511	0.976	5.533	1.155	0.637
VVR-Easy	0.283	0.729	0.268	0.587	0.849	0.294	8.275	0.979	5.483	1.114	0.641
Δ add VVR-Easy	+0.180	+0.030	+0.025	+0.030	+0.0004	+0.005	+0.222	+0.002	+0.009	+0.045	+0.002
VVR-Matched	0.466	0.709	0.360	0.536	0.847	0.291	7.997	0.980	5.472	1.109	0.636
GenEval2 + VVR-Matched	0.335	0.712	0.491	0.510	0.848	0.301	8.498	0.972	5.516	1.148	0.637
Δ add VVR-Matched	+0.296	+0.024	+0.038	+0.009	−0.0002	+0.004	+0.209	−0.001	−0.007	+0.038	+0.001
OCR	0.036	0.625	0.225	0.962	0.844	0.290	7.601	0.960	5.477	0.994	0.633
OCR + VVR-Easy	0.247	0.678	0.252	0.941	0.846	0.290	7.765	0.969	5.488	1.087	0.636
Δ add VVR-Easy	+0.212	+0.053	+0.027	−0.021	+0.002	0.000	+0.164	+0.009	+0.012	+0.092	+0.002
Five-reward	0.048	0.739	0.342	0.856	0.850	0.294	8.137	0.974	5.510	1.125	0.640
Five-reward + VVR-Easy	0.158	0.751	0.383	0.826	0.848	0.300	8.444	0.976	5.541	1.152	0.640
Δ add VVR-Easy	+0.110	+0.012	+0.041	−0.030	−0.002	+0.006	+0.307	+0.002	+0.031	+0.027	+0.0003

Table 16 gives paired bootstrap intervals for the mixture comparisons. The pretrained model's evaluation retained only aggregate scores, so comparisons with it have no intervals.

Table 16: Effect of adding VVR to an existing reward, with paired bootstrap 95% intervals over prompts (10,000 resamples; stratified by prompt set for macro-averaged metrics and by domain for HPSv2.1). The GenEval, GenEval2, OCR, HPSv3, and UnifiedReward evaluations retained only aggregate scores, so they have no intervals.

Contrast	PickScore	HPSv2.1	CLIPScore	Aesthetic	ImageReward
GenEval2 + VVR-Easy − GenEval2	$+0.0004$ $[-0.0012, 0.0021]$	$+0.0047$ $[0.0042, 0.0052]$	$+0.0023$ $[-0.0014, 0.0058]$	$+0.009$ $[-0.003, 0.022]$	$+0.045$ $[0.025, 0.066]$
GenEval2 + VVR-Matched − GenEval2	-0.0002 $[-0.0019, 0.0015]$	$+0.0039$ $[0.0034, 0.0045]$	-0.0011 $[-0.0044, 0.0021]$	-0.007 $[-0.020, 0.006]$	$+0.038$ $[0.020, 0.057]$
OCR + VVR-Easy − OCR	$+0.0018$ $[-0.0002, 0.0037]$	0.0000 $[-0.0007, 0.0006]$	$+0.0086$ $[0.0054, 0.0118]$	$+0.012$ $[-0.001, 0.025]$	$+0.092$ $[0.071, 0.113]$
Five-reward + VVR-Easy − Five-reward	-0.0024 $[-0.0041, -0.0006]$	$+0.0060$ $[0.0054, 0.0066]$	$+0.0023$ $[-0.0009, 0.0055]$	$+0.031$ $[0.020, 0.042]$	$+0.027$ $[0.009, 0.046]$

Matched-complexity GenEval2 mixture. Replacing VVR-Easy with VVR-Matched in the GenEval2 mixture raises VVRBENCH accuracy from 21.82% to 33.50% and GenEval2 from 0.478 to 0.491; accuracy in C_3, C_4, and C_5 rises from 14.08%, 6.41%, and 1.10% to 32.34%, 20.32%, and 9.22%. This mixture raises seven of ten non-VVR metrics over GenEval2 alone, with intervals excluding zero for HPSv2.1 ($+0.004$) and ImageReward ($+0.038$).

G HUMAN PREFERENCE STUDY

Three annotators each compare the same 400 image pairs and choose the image they prefer given the prompt, with a tie option. The study contains two comparisons, VVR-Easy against the pretrained model and GenEval2 mixed with VVR-Easy against GenEval2, and each of five prompt suites contributes 40 prompts to each comparison. The 400-prompt study uses 80 unique prompts from each of VVR, GenEval2, GenEval, OCR, and DrawBench. The VVR prompts were drawn, 16 from each of five complexity bins, from a candidate pool of 11,250 tasks that preceded the final benchmark; 74 of them are VVRBENCH tasks, and none appears in VVR-Easy or Challenge. The GenEval sample is balanced across its six task categories. Each prompt appears in one comparison, paired generations share a sampling seed, and model identity and left and right order are hidden during annotation. Each annotator sees the pairs in an independently randomized order and left-right assignment. Win rates average each prompt's score over the annotators (win 1, tie 0.5, loss 0), and intervals are 95% bootstrap intervals over prompts. Each annotator separately favors the VVR-trained model in every suite of both comparisons. On pairs where both annotators chose an image, the mean pairwise agreement is 83.8% (84.9%, 81.9%, and 84.6% for the three annotator pairs), and Fleiss' κ among the three annotators, with ties as a third label, is 0.49. Table 17 gives the agreement of each annotator pair.

Table 17: Agreement between annotator pairs on the 400 preference pairs. Labels are decoded to the preferred model before comparison. "Both chose" uses only the pairs on which neither annotator chose a tie; "ties as a label" uses all 400 pairs with tie as a third label, which is also the label set of Cohen's κ. Fleiss' κ over the three annotators is 0.49.

Annotator pair	Agreement, both chose (%)	Agreement, ties as a label (%)	Cohen's κ
1 vs. 2	84.9 (298/351)	76.0 (304/400)	0.50
1 vs. 3	81.9 (276/337)	72.0 (288/400)	0.44
2 vs. 3	84.6 (312/369)	78.8 (315/400)	0.53
Mean	83.8	75.6	0.49