VICO: Visual Environments Co-Evolving for Vision-Language Model Reasoning

Meng Lu[1], Ligeng Zhu[2], Olivia Xiao[3], Yuchen Zhuang[4], Zihan Wang[5], Kuncheng Wu[6], Bangya Liu[5], Yu Wang[7], Charles Fleming[8,*], Wenqi Shi[9,*], Xuan Wang[1,*]

[1]Virginia Tech [2]NVIDIA [3]UChicago [4]Georgia Tech
[5]Abaka AI [6]UC Berkeley [7]UGA [8]Cisco [9]WashU

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become a standard recipe for post-training vision-language models (VLMs), but it typically assumes a static training environment. As the actor improves, fixed tasks drift out of its learning frontier: many become trivial, others remain unsolvable; and the learning signal collapses. We argue that *VLM post-training should evolve the visual environment alongside the actor*, not just the actor itself. We propose VICO, a co-evolutionary framework in which an *actor* and an *Environment-as-Rewriter* (EnvRewriter) are trained jointly: the EnvRewriter edits verifiable image-side structures, such as scene graphs, chart tables, or protected region masks, and re-renders them to produce label-valid training samples whose difficulty is calibrated to the actor's current ability through a pass-rate-based reward. This loop continuously realigns task difficulty with actor capability without any additional human annotation. Across nine multimodal benchmarks spanning mathematical reasoning and visually grounded understanding, VICO-8B improves over its base model by up to **+5.0%** on out-of-domain tasks, surpasses the strongest self-evolution and text-editing co-evolution baselines by **+4.3%** and **+8.4%** respectively, and stays comparable to chart-specialized RLVR methods using **16–160× fewer labeled samples**. By shifting from human-labeled supervision to image-editing co-evolution, VICO offers a scalable path beyond static-corpus RLVR for visual reasoning. GitHub: https://github.com/Lucany/VICO

1 Introduction

Large Vision-Language Models (VLMs) have recently improved through reinforcement learning with verifiable rewards (RLVR) [13], where models learn from automatically checkable outcomes rather than dense human rationales [33]. This recipe has made self-evolution an attractive paradigm: rather than relying on costly human supervision, a model can generate tasks, collect rollouts, and improve from its own experience [22, 39]. Yet most self-evolution pipelines share

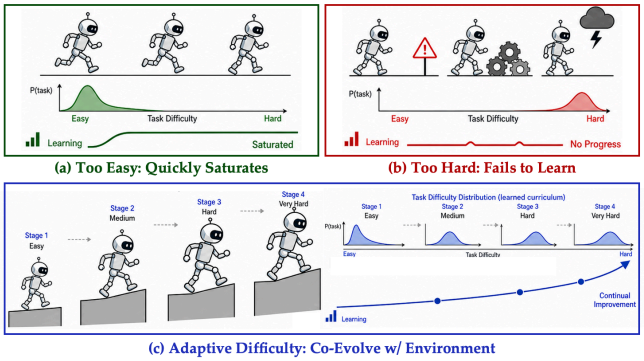


Figure 1: **Co-evolution with adaptive visual environments and actors leads to better training.**

*Equal correspondence to: Charles Fleming, Wenqi Shi, Xuan Wang.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a structural limitation: the actor keeps changing, but the visual environment stays fixed. As the actor improves, fixed tasks drift out of its learning frontier, where many become trivial while others remain unsolvable. In both regimes, the rollout group provides little useful contrast (Figure 1): the empirical pass rate $\hat{p}$ moves toward 1 or 0, the Group Relative Policy Optimization (GRPO) reward variance, proportional to $\hat{p}(1-\hat{p})$, collapses, and the learning signal weakens precisely when the actor is ready to learn more [14]. Crucially, $\hat{p}$ already tells us exactly how difficult each task is for the current actor, yet existing methods use it only for normalization or filtering, discarding a signal that could otherwise guide the environment itself. This raises the central question of our work: *can the visual environment adapt to the actor using the same rollout feedback that trains the actor?*

Answering this question (*i.e.*, adapting the visual environment), however, requires more than simply generating new images. Useful training samples must satisfy two constraints simultaneously. First, they should be *verifiable*: edits should not introduce ambiguous questions or incorrect labels, otherwise the verifiable reward signal that underpins RLVR breaks down. Second, they should be *capability-aligned*: edits should move tasks toward the actor's current learning frontier rather than making them arbitrarily harder or easier. Human annotation [28, 27] satisfies both but does not scale, while offline augmentation with stronger teachers [5, 24] reduces annotation cost but produces a static distribution that cannot track the actor's evolving weaknesses. What is missing is an automatic visual environment that adapts to the actor while preserving the validity of the label.

We propose VICO, a co-evolution framework that addresses both requirements through joint training of two VLMs. The *actor* learns to answer visual reasoning questions, while the *EnvRewriter* learns to perform verifiable structured edits to match the actor's current ability. For each rewritten task, we sample a group of actor rollouts and reuse the resulting pass rate $\hat{p}$ as a shared learning signal: binary rollout rewards update the actor through GRPO, while $\hat{p}$ updates the EnvRewriter through an α-curriculum reward that peaks near $\hat{p} = 0.5$, steering edits into the actor's *learnable zone*. We then evaluate VICO across three representative visual reasoning domains, including compositional reasoning, chart understanding, and general visual question answering, and against three categories of baselines: text-only rewriting, static-environment self-evolution, and offline augmentation with stronger teachers, including GPT-5 [31] and Nano Banana [11]. Across eight out-of-distribution benchmarks, VICO exceeds the strongest self-evolution baseline by +4.3% on average. Notably, even when stronger image generators are scaled up, static baselines plateau, indicating that the gains stem from the iterative actor–environment loop rather than from data generation quality alone. Our contributions are as follows:

- **An actor–environment co-evolution paradigm for VLM reasoning.** We introduce a framework in which the visual environment evolves alongside the actor, using the rollout pass rate $\hat{p}$ as a shared learning signal. This continually realigns task difficulty with actor capability and overcomes the saturation that limits fixed-environment self-evolution.
- **A verifiable image-editing training framework.** We design a pipeline that couples structured visual editing with RLVR, converting edited images into valid training samples through deterministic rendering and programmatic relabeling, with no human annotation.
- **Strong empirical performance across diverse benchmarks.** Across nine multimodal benchmarks, VICO surpasses RLVR baselines by +15.5% on average, exceeds the strongest self-evolution baseline by +4.3% on out-of-domain tasks, and outperforms the proprietary Gemini-2.5-Pro by +2.5%.
- **Annotation-efficient training.** VICO is competitive with chart-specialized RLVR methods using 16×–160× fewer labeled samples, demonstrating that adaptive environments can substitute for large-scale human or teacher-model annotation.

2 Related Work

Reinforcement Learning for Vision-Language Reasoning. RLVR has become a standard post-training recipe for language and vision-language models, replacing dense human rationales with rule- or program-checkable outcomes [21, 33, 25]. Recent multimodal extensions of RLVR target multi-step visual reasoning through cross-modal symbolic formalization [49], scaled rule-based reward training [29], and combined SFT-RL pipelines [9, 26, 46]. Subsequent work refines RLVR through efficient visual processing [48], weakness-driven problem generation [23], curiosity-driven

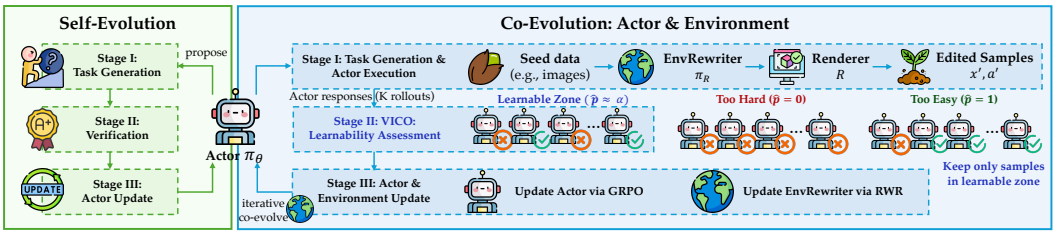


Figure 2: **VICO turns self-evolution into an adaptive actor–environment loop.** The EnvRewriter creates visual tasks, the Actor solves them, and shared rollouts update both policies: task rewards train the Actor, while difficulty-aligned rewards train the EnvRewriter.

exploration [8], and multi-turn world-model reasoning [38]. While these methods differ in reward design, exploration strategy, and training schedule, they share a common assumption: a fixed pool of visual tasks. *VICO instead treats the visual environment as a learnable component that co-adapts with the actor*, aiming to unlock new capabilities rather than refine existing ones over a static distribution [52].

Self-Evolving VLMs. Self-evolving VLMs reduce reliance on human-curated supervision by letting models generate, solve, or refine their own training data. MM-Zero bootstraps multimodal reasoning from zero labeled data by proposing and solving visual problems [22], VisPlay self-generates question-answer pairs from raw images for RL training [16], and Vision-Zero frames self-improvement as visual self-play with verifiable outcomes [39]. Related efforts apply self-evolution to visual games [45, 35], perception [54], image-quality assessment [42], self-refinement [10], and OCR hallucination mitigation [17]. Across these methods, the actor evolves but the visual environment remains fixed, leaving the training distribution increasingly misaligned with the actor's current ability.

Co-Evolving VLMs and Environments. A more recent line of co-evolution work allows the environment to change with the model: GenEnv trains an environment policy for difficulty-aligned planning tasks [14], V-Zero perturbs the textual side of visual tasks while keeping the image pool fixed [36], C2-Evo co-evolves multimodal data and model training through model-driven generation [7], and Active-Zero retrieves from a large external image pool in an offline augmentation setting [15]. VICO differs from this line in two ways: it edits the visual substructure of each image rather than perturbing text or retrieving from a fixed pool, directly adapting the modality where reasoning is grounded; and it uses the actor's own pass rate as the supervision for environment updates, providing a difficulty-calibration loop tightly coupled to the actor's learning trajectory.

3 Methodology

VICO views visual reasoning training as an *actor-environment curriculum process* rather than a single model self-optimization problem. We maintain two policies: an *actor* π_a (the vision-language model being trained) and an *EnvRewriter* π_R (the environment that produces training samples). Unlike standard RLVR with a fixed training source, VICO enables both to co-evolve: π_a learns to solve visual reasoning tasks whose difficulty is shaped by the current EnvRewriter, while π_R is rewarded for producing edited images whose difficulty is aligned with the actor's current capability — targeting the "learnable zone" where the learning signal is most effective.

3.1 Data-Evolving Paradigm: From Static RLVR to Adaptive Co-evolution

Standard RLVR minimizes a loss $\mathcal{L}(\theta)$ over a static dataset $\mathcal{D}_{\text{static}} = \{(x, q, a)\}$, where θ denotes the parameters of the actor. In contrast, VICO implements a *Data-Evolving Paradigm*. At each round t, the training dataset $\mathcal{D}_t$ is generated dynamically by the EnvRewriter π_R, based on the actor's current pass rate. This establishes a feedback loop (Figure 2) in which the EnvRewriter's goal is not to beat the actor, but to identify its "breaking points" and convert them into a training signal.

Table 1: **VICO keeps edited samples verifiable across domains.** Structured domains use deterministic recomputation, while natural images use constrained answer preservation; neither needs judging or relabeling.

Domain	Editing Interface	Label-Validity Rule
CLEVR [19]	Edit the scene graph: attributes, objects, shapes, or relations.	Re-execute the question program on the edited graph; reject rendering or ambiguity failures.
ChartQA [28]	Edit the CSV table under a two-tier rule: Tier 1 has a unique answer cell; Tier 2 has no unique cell mapping.	Tier 1 computes $a' = \text{table}'[r, c]$; Tier 2 preserves $a' = a$. Reject schema, rendering, or invariance failures.
ZwZ [41]	Edit only the complement of protected target masks, e.g., background or non-target distractors.	Preserve $(q', a') = (q, a)$; reject edits whose mask overlaps protected targets or fails rendering.

3.2 Seed Visual Data and Verifiable Sample Generation

We start VICO from a small seed visual dataset and let the EnvRewriter derive additional training examples by editing the structured or protected representation of each seed sample in subsequent rounds. The key requirement is that every edited sample remains *label-valid* while its difficulty changes. We therefore avoid unconstrained pixel-level generation. Instead, the answer can either be recomputed programmatically from the edited structure or preserved by construction through constrained editing. Table 1 summarizes the editing interface for each domain.

We experiment with three types of data: $2{,}000$ CLEVR with scene graphs and question programs, $2{,}000$ ChartQA with their underlying CSV tables, and $1{,}000$ ZwZ with answer-relevant region annotations. The actor–environment co-evolution loop is shared across all three domains; only the verifiable editing interface depends on the structure available in each dataset. More details are provided in Appendix C. We further analyze seed-sampling stability in Appendix H.5; three ChartQA runs with different seed samples fluctuate by less than 2%.

3.3 Rewards and Co-Evolutionary Optimization

Actor Reward and Update. The EnvRewriter generates a training instance (x', q', a') with a verified answer a', where a' is programmatically computed from the modified structured representation (Appendix C). This verifiable answer enables rule-based rewards, avoiding neural reward models that are prone to reward hacking. The actor's response receives a reward $+1$ if it both produces the correct answer (exact match against a') and follows the required `<think>`/`<answer>` format; otherwise, it receives -1 (Appendix B.4) [57]. The actor π_a is updated to maximize this reward via GRPO [33] on edited triples sampled from $\mathcal{D}_a$.

EnvRewriter Reward and Update. The core innovation for π_R is a difficulty-calibration reward that targets a success-rate band around $\alpha = 0.5$. For each edited image x', the actor generates K rollouts, and we compute the empirical pass rate $\hat{p}_{\text{final}}$ as the fraction of correct rollouts. The EnvRewriter reward is a bell-shaped function of $\hat{p}_{\text{final}}$ peaked at α (Appendix B.4), which discourages edits that are already mastered ($\hat{p}_{\text{final}} \to 1$) or hopeless ($\hat{p}_{\text{final}} \to 0$). We additionally apply a difficulty filter that excludes edits with $|\hat{p}_{\text{final}} - \alpha| > k_{\min}$ from EnvRewriter updates, where $k_{\min} \in (0, \frac{1}{2}]$ is the maximum allowed deviation from α, to prevent overfitting to transient spikes; a theoretical justification is provided in Section 4. The EnvRewriter π_R is updated via Reward-Weighted Regression (RWR) on a weighted SFT set of edit trajectories, with stability ensured by a KL penalty to the initial EnvRewriter $\pi_R^{(0)}$ and a maximum KL threshold $\delta_{\max}$. Appendix B.1 Algorithm 1 summarizes the full optimization loop.

4 Theoretical Analysis of Difficulty-Calibrated VICO

We provide a simple analysis to clarify two properties of VICO's co-evolution mechanism. First, edits with pass rates close to $\alpha = 0.5$ provide the strongest learning signal for the actor. Second, as the actor matures, the optimal target can shift away from 0.5, which explains the saturation observed in our experiments. Full derivations are deferred to Appendix E.

Intermediate difficulty maximizes the actor's learning signal. Consider a stylized bandit setting where the actor receives a binary reward $r \in \{0, 1\}$ with success probability p on an edited image x'. Under a REINFORCE-style update [43] with a variance-minimizing baseline [20], and under a standard bounded-score assumption for policy-gradient updates, the squared gradient norm satisfies

$$\mathbb{E}\big[\|g(x', r)\|^2\big] \;\propto\; p(1-p), \tag{4.1}$$

up to constants independent of p. Since $p(1-p)$ is concave and maximized at $p = 1/2$, the gradient signal is strongest for edits of intermediate difficulty. Equivalently, maximizing $p(1-p)$ is the same as minimizing $(p - \frac{1}{2})^2$. This supports adopting the α-Curriculum reward [14] $R_{\text{rewriter}}(\hat{p}) = \exp\big(-\beta(\hat{p} - \alpha)^2\big)$ with $\alpha = 0.5$ as the difficulty-calibration objective for the EnvRewriter.

A drift-corrected target predicts saturation. The target $\alpha = 0.5$ is optimal only when edits are costless. As the actor improves, its pass rate $\hat{p}_{\text{orig},t}$ on the unedited image rises. Forcing the edited sample back to $\hat{p} = 0.5$ may then require increasingly aggressive edits. These edits can move the training distribution away from the natural image manifold [1]. We model this trade-off by adding a quadratic drift penalty:

$$\Delta\text{Acc}_t = C_t p(1-p) - \lambda\kappa_t (p - \hat{p}_{\text{orig},t})^2, \tag{4.2}$$

where $C_t > 0$ denotes the actor's remaining learning headroom, and $\lambda\kappa_t \geq 0$ controls the drift cost. Maximizing this concave objective gives

$$p_t^* = \tfrac{1}{2} + \rho_t \left(\hat{p}_{\text{orig},t} - \tfrac{1}{2}\right), \quad \rho_t = \frac{\lambda\kappa_t}{C_t + \lambda\kappa_t} \in [0, 1], \tag{4.3}$$

so the optimal target is a convex combination of 0.5 and $\hat{p}_{\text{orig},t}$. When the actor has large headroom ($\rho_t \to 0$), the optimal target remains near 0.5. As headroom shrinks or drift cost increases ($\rho_t \to 1$), the optimum moves toward the unedited pass rate. Continued editing then enters a drift-dominated regime, where stronger perturbations bring smaller gains and may eventually hurt performance. This analysis predicts diminishing per-round gains, fewer sweet-zone samples, and eventual saturation, which we verify in Appendix E.6.

5 Experiments

5.1 Experimental Setups

Training Setup. For the Actor π_a, we train via GRPO [33] with $n = 8$ on-policy rollouts and a learning rate 1×10^{-6}. The EnvRewriter π_R is updated via reward-weighted LoRA SFT with $\beta = 4.0$ and $\alpha = 0.5$ ([14]). We use AdamW for both policies, run VICO for $T = 5$ rounds, and use up to $8\times$ A100-80GB GPUs with verl [34]. We train Qwen3-VL-8B [2] on three visual datasets, and Qwen2.5-VL-7B [3] and InternVL3-8B [58] on chart understanding to assess backbone generalizability. We validate VICO on three visual reasoning domains: CLEVR [19] for compositional reasoning, ChartQA [28] for chart reasoning, and ZwZ [41] for natural-image VQA. Full hyperparameters, two-stage training, dataset examples, and multi-seed analyzes are in Appendix B.

Tasks and Datasets. We evaluate on nine benchmarks under identical zero-shot settings (temperature 0.7, top-p 1.0, max response length 3,824 tokens). For mathematical reasoning, we use MathVista [27], MathVision [37], MathVerse [55], WeMath [32], and LogicVista [44]. For chart reasoning, we use ChartQA [28]. For general vision-centric understanding, we use MMMU [51], MMVP [53], and BLINK [12]. The *Average* column in Table 2 is the equally weighted average of the mathematical-reasoning average and the vision-centric average.

Baselines. We compare VICO against three categories of strong baselines reflecting different post-training paradigms: (1) **RLVR-trained reasoning models** trained with RLVR on human-labeled data—R1-OneVision-7B [49], MM-Eureka-Qwen-7B [29], OpenVLThinker-7B [9]; (2) **Self-evolution models** that evolve the model in a fixed environment—MM-Zero-8B [22], three VisionZero-7B variants (CLEVR, Chart, Real-world), and VisionZero-InternVL3-8B [39]; (3) **Co-evolution baseline** V-Zero [36], which perturbs question-answer pairs without editing the visual environment. These three categories isolate the effect of *visual-environment co-evolution* from data-rich RL post-training and model-side self-evolution. Implementation details are in Appendix B.

Table 2: **VICO achieves the strongest overall performance among comparable open-source VLMs.** We report accuracy on nine multimodal reasoning benchmarks; **bold** marks the best result and underline marks the second-best among non- commercial models.

Baselines (↓)	**Mathematical Reasoning**							**General Vision-Centric**				**All**
Datasets (→)	MathVista	MathVision	ChartQA	MathVerse	WeMath	LogicVista	avg.	MMMU	MMVP	BLINK	avg.	avg.
Commercial VLMs (For reference)												
GPT-4o	61.4	30.4	80.7	50.2	40.0	45.9	51.4	65.1	86.3	60.0	70.47	61.0
GPT-5	80.2	41.1	85.9	68.1	64.7	49.1	64.9	68.6	84.7	74.8	76.0	70.4
Gemini-2-Flash	73.4	41.3	77.0	54.4	57.1	56.2	59.9	52.6	83.0	59.6	65.1	62.5
Gemini-2.5-Pro	72.8	29.6	80.1	67.0	68.9	52.7	61.9	54.8	75.7	78.1	69.5	65.7
Gemini-3.1-pro	83.3	34.7	84.1	67.8	65.9	57.4	65.5	60.0	89.7	83.5	77.7	71.6
Open-Source VLMs												
Qwen2.5-VL-7B	68.2	25.6	76.4	39.6	60.7	47.2	53.0	54.7	76.8	55.2	62.2	57.6
Qwen3-VL-8B	67.7	31.5	76.9	50.6	72.0	55.1	59.0	55.8	77.7	68.9	67.5	63.2
InternVL3-8B	60.4	21.3	69.9	32.2	61.3	34.6	46.6	55.3	64.3	61.2	60.3	53.5
InternVL3-14B	74.1	33.8	87.3	44.2	68.9	51.6	60.0	56.4	76.30	60.2	64.3	62.2
RL-Trained Reasoning Baselines												
OpenVLThinker-7B	70.2	25.3	59.9	47.9	36.5	44.3	47.4	36.1	71.7	61.2	56.3	51.8
MM-Eureka-Qwen-7B	73.0	26.9	58.2	50.3	36.2	42.9	47.9	50.1	73.0	58.3	60.5	54.2
MM-Eureka-InternVL-8B	62.4	22.1	54.9	32.1	58.2	38.9	44.8	51.6	74.3	61.0	62.3	53.5
R1-OneVision-7B	64.1	24.1	73.4	46.4	61.8	44.5	52.4	46.4	61.3	48.7	52.1	52.3
Self-evolution												
MM-Zero (Qwen3-VL-8B)	67.7	39.6	80.8	45.1	73.4	55.4	60.3	58.7	77.3	67.5	67.8	64.1
VisionZero-Qwen2.5-VL-7B (CLEVR)	72.2	28.4	86.3	53.2	49.8	45.8	56.0	53.0	79.5	56.0	62.8	59.4
VisionZero-Qwen2.5-VL-7B (Chart)	72.4	28.9	87.2	52.5	51.2	43.5	56.0	53.6	79.1	56.8	63.2	59.6
VisionZero-Qwen2.5-VL-7B (RW)	73.1	28.5	86.3	52.1	50.8	42.6	55.6	52.5	79.2	57.2	63.0	59.3
VisionZero-InternVL3-8B	62.2	24.2	53.0	32.9	28.7	41.8	40.5	56.9	69.0	63.3	63.1	51.8
Co-evolution												
V-Zero (Qwen2.5-VL-7B)	69.6	27.0	-	42.6	-	48.6	47.0	58.6	-	66.9	62.8	54.9
VICO-Qwen3-VL-8B (CLEVR)	72.8	**41.0**	86.3	**56.0**	**76.8**	57.8	65.1	64.9	**80.3**	69.9	71.7	68.4
Direct GRPO	71.6$_{-1.2\downarrow}$	37.8$_{-3.2\downarrow}$	78.1$_{-8.2\downarrow}$	54.0$_{-2.0\downarrow}$	71.2$_{-5.6\downarrow}$	54.7$_{-3.1\downarrow}$	61.2$_{-3.9\downarrow}$	64.1$_{-0.8\downarrow}$	78.3$_{-2.0\downarrow}$	66.1$_{-3.8\downarrow}$	69.5$_{-2.2\downarrow}$	65.4$_{-3.0\downarrow}$
Supervised GRPO	69.4$_{-3.4\downarrow}$	36.9$_{-4.1\downarrow}$	78.6$_{-7.7\downarrow}$	54.1$_{-1.9\downarrow}$	72.5$_{-4.3\downarrow}$	56.0$_{-1.8\downarrow}$	61.2$_{-3.9\downarrow}$	54.6$_{-10.3\downarrow}$	76.3$_{-4.0\downarrow}$	68.3$_{-1.6\downarrow}$	66.4$_{-5.3\downarrow}$	63.8$_{-4.6\downarrow}$
VICO-Qwen3-VL-8B (Chart)	**78.4**	39.4	**89.1**	55.0	74.5	56.5	**65.5**	**65.2**	79.7	70.5	**71.8**	**68.6**
Direct GRPO	70.9$_{-7.5\downarrow}$	37.9$_{-1.5\downarrow}$	83.8$_{-5.3\downarrow}$	52.7$_{-2.3\downarrow}$	73.0$_{-1.5\downarrow}$	57.0$_{+0.5\uparrow}$	62.5$_{-3.0\downarrow}$	65.0$_{-0.2\downarrow}$	77.7$_{-2.0\downarrow}$	65.8$_{-4.7\downarrow}$	69.5$_{-2.3\downarrow}$	66.0$_{-2.6\downarrow}$
Supervised GRPO	75.6$_{-2.8\downarrow}$	36.1$_{-3.3\downarrow}$	79.4$_{-9.7\downarrow}$	51.8$_{-3.2\downarrow}$	70.1$_{-4.4\downarrow}$	56.0$_{-0.5\downarrow}$	61.5$_{-4.0\downarrow}$	56.8$_{-8.4\downarrow}$	76.7$_{-3.0\downarrow}$	68.7$_{-1.8\downarrow}$	67.4$_{-4.4\downarrow}$	64.4$_{-4.2\downarrow}$
VICO-Qwen3-VL-8B (Real VQA)	75.8	40.8	86.2	54.1	73.2	**59.6**	65.0	61.5	79.3	**71.5**	70.8	67.9
Direct GRPO	71.9$_{-3.9\downarrow}$	37.1$_{-3.7\downarrow}$	79.8$_{-6.4\downarrow}$	51.0$_{-3.1\downarrow}$	72.1$_{-1.1\downarrow}$	58.3$_{-1.3\downarrow}$	61.7$_{-3.3\downarrow}$	53.7$_{-7.8\downarrow}$	77.7$_{-1.6\downarrow}$	57.2$_{-14.3\downarrow}$	62.9$_{-7.9\downarrow}$	62.3$_{-5.6\downarrow}$
Supervised GRPO	70.6$_{-5.2\downarrow}$	37.6$_{-3.2\downarrow}$	77.0$_{-9.2\downarrow}$	53.0$_{-1.1\downarrow}$	71.2$_{-2.0\downarrow}$	58.9$_{-0.7\downarrow}$	61.4$_{-3.6\downarrow}$	54.2$_{-7.3\downarrow}$	76.3$_{-3.0\downarrow}$	58.8$_{-12.7\downarrow}$	63.1$_{-7.7\downarrow}$	62.2$_{-5.7\downarrow}$

Table 3: **VICO improves VLM backbones with far less training data.** The same pipeline improves Qwen2.5-VL-7B and InternVL3-8B, while using 16×–160× less data than chart-specialized RL models. Overall average uses the seven displayed benchmarks; bold marks the best result and underline marks the second-best. **Average is computed over reported columns only.*

Baselines (↓)	**Seed Data**	**Mathematical Reasoning**					**Vision-Centric**		**Overall**
Datasets (→)		MathVista	MathVerse	ChartQA	WeMath	LogicVista	MMMU	BLINK	avg.
Chart-Specialized Methods									
Chart-R1-7B [6]	∼33k	-	-	**91.0**	-	-	-	-	-
Bespoke-MiniChart-7B [30]	∼91k	-	-	89.5	-	-	-	-	-
ECD (Qwen2.5-VL-7B) [50]	∼320k	-	-	85.3	-	-	-	-	-
Qwen2.5-VL-7B Family									
Qwen2.5-VL-7B (base)	–	68.2	39.6	76.4	60.7	**47.2**	**54.7**	55.2	57.4
OpenVLThinker-7B [9]	∼50k	70.2	47.9	59.9	36.5	44.3	36.1	**61.2**	50.9
MM-Eureka-Qwen-7B [29]	15k	73.0	50.3	58.2	36.2	42.9	50.1	58.3	52.7
R1-OneVision-7B [49]	10k	64.1	46.4	73.4	61.8	44.5	46.4	48.7	55.0
VisionZero-Qwen7B (CLEVR) [39]	2000	72.2	53.2	86.3	49.8	45.8	53.0	56.0	59.5
VisionZero-Qwen7B (Chart)	1000	72.4	52.5	87.2	51.2	43.5	53.6	56.8	59.6
VisionZero-Qwen7B (Real-world)	1000	**73.1**	52.1	86.3	50.8	42.6	52.5	57.2	59.2
C^2-Evo [7]	–	50.2	19.1	-	-	-	-	-	34.7*
VICO-Qwen2.5-VL-7B (Chart)	2000	72.8	**54.9**	**88.0**	**67.6**	44.0	53.8	53.7	**62.1**
InternVL3-8B Family									
InternVL3-8B (base)	–	60.4	32.2	69.9	61.3	34.6	55.3	61.2	53.6
MM-Eureka-InternVL-8B	15k	62.4	32.1	54.9	58.2	38.9	51.6	61.0	51.3
VisionZero-InternVL3-8B	2000	62.2	32.9	53.0	28.7	**41.8**	**56.9**	**63.3**	48.4
VICO-InternVL3-8B (Chart)	2000	**68.9**	**57.6**	**80.6**	**63.4**	40.2	55.7	58.5	**60.7**

5.2 Main Results

VICO-actor consistently improves over the base model and outperforms three categories of baselines: RL-trained reasoning models, self- and co-evolution methods. Table 2 presents the main comparison. For the Chart Understanding variant, we report the OOD average over the eight benchmarks excluding ChartQA, which is its in-domain benchmark. For the CLEVR and ZwZ variants, all nine evaluation benchmarks are treated as out-of-domain. Notably, VICO attains these improvements using only 2000 seed inputs, which is 16×–160× fewer than the chart-specialized RLVR baselines that depend on large amounts of human or model-generated CoT annotations (Table 3). The strongest variant, VICO-8B (Chart), achieves the best OOD average accuracy of 64.9% (excluding in-domain ChartQA), improving over the base model by +5.0%. Against RL-trained reasoning baselines, VICO surpasses them by +15.5% on average. Against the strongest

Table 4: **Visual edits outperform text-only perturbations.** With the same CLEVR seed source, image editing in VICO improves the overall average by $+6.1\%$, compared with $+2.4\%$ for text edits. The largest gap appears on ChartQA ($+9.4\%$ vs. $+0.9\%$).

Benchmark	Base	VICO-Qwen3VL-8B (CLEVR, text-edit)	VICO-Qwen3VL-8B (CLEVR, image-edit)
Mathematical reasoning			
MathVista	67.7	$70.3_{+2.6\uparrow}$	**72.8**$_{+5.1\uparrow}$
MathVision	31.5	$39.0_{+7.5\uparrow}$	**41.0**$_{+9.5\uparrow}$
MathVerse	50.6	$53.9_{+3.3\uparrow}$	**56.0**$_{+5.4\uparrow}$
WeMath	72.0	$73.1_{+1.1\uparrow}$	**76.8**$_{+4.8\uparrow}$
LogicVista	55.1	$54.1_{-1.0\downarrow}$	**57.8**$_{+2.7\uparrow}$
Chart reasoning			
ChartQA	76.9	$77.8_{+0.9\uparrow}$	**86.3**$_{+9.4\uparrow}$
Overall avg.	59.0	$61.4_{+2.4\uparrow}$	**65.1**$_{+6.1\uparrow}$

Table 5: **Iterative co-evolution beats static augmentation.** Strong offline generators such as GPT-5 and Nano Banana improve ChartQA accuracy, but they remain below actor-conditioned visual co-evolution, which reaches 89.1%.

Setting	Iter. loop	Visual edit	ChartQA
Base (Qwen3-VL-8B)	–	–	76.9
Static offline augmentation (no loop)			
Offline-Random (self-gen)	✗	✓	$78.6_{+1.7\uparrow}$
Offline-GPT-5	✗	✓	$80.1_{+3.2\uparrow}$
Offline Nano Banana	✗	✓	$81.9_{+5.0\uparrow}$
RLVR-only (no environment adaptation)			
Supervised GRPO	✗	✗	$79.4_{+2.5\uparrow}$
Direct GRPO	✗	✗	$83.8_{+6.9\uparrow}$
VICO-Qwen3VL-8B (Chart, text-edit)	✓	✗	$77.2_{+0.3\uparrow}$
VICO-Qwen3VL-8B (Chart, image-edit)	✓	✓	**89.1**$_{+12.2\uparrow}$

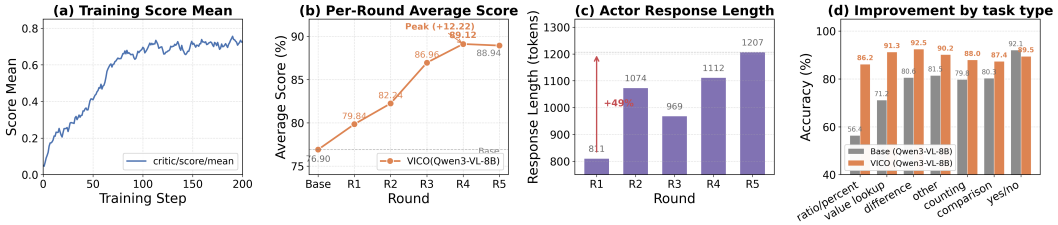


Figure 3: **VICO trains stably and induces an emergent curriculum.** (a) GRPO reward improves smoothly. (b) ChartQA accuracy peaks at Round 4 with a $+12.2$ gain. (c) Actor responses grow by $+49\%$, suggesting harder tasks and longer reasoning. (d) Gains concentrate on reasoning-heavy question types.

self-evolution baseline (MM-Zero), the three VICO variants surpass it by $+4.3\%$ on the OOD benchmarks. Against the text-editing co-evolution baseline (V-Zero), VICO outperforms it by $+8.4\%$ on the overlapping benchmarks. Beyond open-source baselines, VICO-8B also outperforms Gemini-2-Flash and Gemini-2.5-Pro by $+5.2\%$ and $+2.5\%$ on the OOD benchmarks, showing that an 8B model trained through co-evolution can compete with much larger proprietary VLMs.

VICO has strong generalization across OOD domains and models. As shown in Table 2, VICO-8B variants gain $+5.2\%$ on average across OOD benchmarks. The transfer is strongest in mathematical reasoning: without math-specific training, VICO-8B (CLEVR), VICO-8B (Chart), and VICO-8B (ZwZ) improve over Qwen3-VL-8B by $+5.5\%$, $+5.4\%$, and $+5.3\%$ on MathVista, MathVision, MathVerse, WeMath, and LogicVista. This suggests that VICO learns transferable reasoning skills rather than domain-specific edit shortcuts. By comparison, existing RLVR baselines transfer less reliably: R1-OneVision-7B and MM-Eureka-Qwen-7B improve on one benchmark, but Qwen2.5-based RLVR baselines drop by -7.3% on average relative to their base models. VICO also transfers across backbones. Table 3 shows that VICO-Qwen2.5-VL-7B (Chart) and VICO-InternVL3-8B (Chart) improve over their base models by $+4.7\%$ and $+7.1\%$ on average.

Editing images is more effective than rewriting text alone. We compare visual-environment editing with text-only perturbation in Table 4. Within the same VICO environment, image editing yields a $+6.1\%$ gain in the evaluated reasoning benchmarks, while text-only perturbation improves by only $+2.4\%$. The gap is especially clear on ChartQA, where image editing gains $+9.4\%$ versus only $+0.9\%$ for text-only perturbation. This finding is further supported by independent evidence: in the separately developed V-Zero framework, text-only perturbation also improves mathematical reasoning by only $+1.8\%$ over its base model. Together, these results show that the key bottleneck is the editing modality itself: effective co-evolution requires modifying the visual environment, not just rewriting the text.

VICO trains stably and improves across training rounds. Figure 3(a) shows that the step-wise GRPO training score increases smoothly without collapse or divergence. Across rounds, Figure 3(b) shows that VICO steadily improves on ChartQA. Accuracy rises from 76.9% at the base model to 89.1% at R4, giving a total gain of $+12.2\%$. Each round contributes a positive gain. This improvement also appears at the sample level. Appendix Table 27 shows that R1 unlocks 13.9% test samples, and the cumulative unlocked fraction grows to 16.1% by R4. These results show that VICO accumulates new correct behaviors across rounds rather than reflecting round-specific fluctuations.

VICO's Gains concentrate on reasoning-heavy question types. Figure 3(d) breaks down ChartQA accuracy by task type. The largest gains appear in multi-step reasoning, including ratio/percentage ($+29.8\%$), value lookup ($+20.1\%$), and difference ($+11.9\%$). In contrast, visually simpler tasks improve less, such as comparison ($+7.1\%$) and counting ($+8.2\%$). The yes/no category slightly drops (-2.6%), likely because it is already saturated. These results suggest that VICO mainly improves reasoning-intensive skills rather than simply boosting easy visual recognition patterns.

VICO turns errors into correct answers with limited regression. To understand where the gain of round 4 comes from, we compare each sample's correctness between the base model and our round 4 actor on the ChartQA test set (Appendix Table 26). VICO converts 365 samples from wrong to right (14.6%), while only 64 samples flip from right to wrong (2.6%). This gives an improvement-to-regression ratio of $\mathbf{5.7}\times$. The net gain of 301 samples corresponds to $+12.0\%$, closely matching the aggregate accuracy gain of $+12.2\%$. Qualitatively, wrong-to-right cases often show explicit step-by-step decomposition, whereas the base model tends to give single-shot answers; examples are provided in Appendix I.6. These results show that the gains mainly come from newly acquired correct behaviors, not from trading off many previously correct samples.

5.3 Co-Evolution Dynamics, Capability Emergence, and Ablations

We then examine whether the actor–environment loop behaves as intended and whether this loop is the source of the gains.

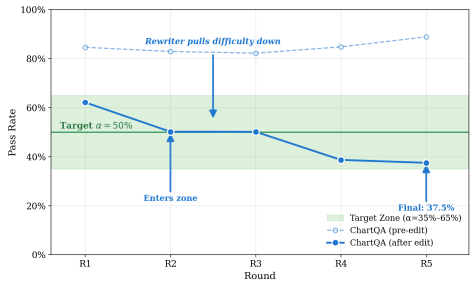
Both Rewriter SFT and the co-evolution loop are required. Table 6 separates the roles of supervised initialization and iterative co-evolution. Actor SFT alone improves ChartQA by only $+2.5\%$, so stronger actor initialization cannot explain the full gain. Running the co-evolution loop with a vanilla EnvRewriter gives only $+1.7\%$, because the Rewriter cannot effectively produce executable edit plans. Thus, the loop alone is also insufficient without a valid environment policy. The full $+12.2\%$ gain appears only when the EnvRewriter is SFT-initialized and then optimized through the iterative actor–environment loop. This shows that Rewriter SFT provides valid environment edits, while co-evolution turns those edits into actor-aligned training signals.

Table 6: **Rewriter initialization is not enough; co-evolution unlocks the full gain.** Actor SFT (w/o co-training) and a vanilla EnvRewriter give modest gains ($+2.5\%$ and $+1.7\%$), while an SFT-initialized EnvRewriter optimized in the actor–environment loop reaches $+12.2\%$.

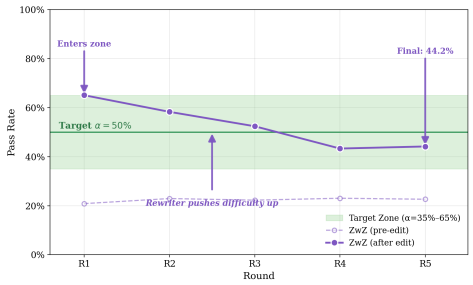
Baselines	**ChartQA (Δ)**
Base (Qwen3-VL-8B)	76.9
Actor SFT only	79.4 $_{+2.5\uparrow}$
Co-evolution (w/ Vanilla Re-writer)	78.6 $_{+1.7\uparrow}$
Co-evolution (VICO)	**89.1** $_{+12.2\uparrow}$

The EnvRewriter learns bidirectional difficulty calibration. We compare two domains with opposite initial actor competence: ChartQA starts easy for the actor ($\hat{p}_{\text{orig}} \approx 0.85$), while ZwZ starts hard ($\hat{p}_{\text{orig}} \approx 0.21$). The EnvRewriter adapts in opposite directions. It makes ChartQA harder and lowers $\hat{p}_{\text{final}}$, while making ZwZ easier and raising $\hat{p}_{\text{final}}$. In both cases, $\hat{p}_{\text{final}}$ enters $[0.35, 0.65]$ within the first few rounds (Figure 4). This shows actor-relative calibration instead of random perturbation.

Stronger edits reflect learned decisions, not longer instructions. We next ask whether the EnvRewriter becomes a stronger editor across rounds. We measure edit intensity as $|\hat{p}_{\text{orig}} - \hat{p}_{\text{final}}|$, the pass-rate shift caused by an edit for the current actor. Edit intensity increases from 0.22 to 0.51, showing that the EnvRewriter changes task difficulty more effectively. This increase is not due to longer outputs: the average instruction length remains nearly unchanged ($973 \rightarrow 981$ tokens) (Figure 10 in the Appendix). In contrast, the actor's response length grows by $+49\%$ (Figure 3(c)), consistent with longer reasoning traces on harder tasks. These trends provide behavioral evidence of co-adaptation: the EnvRewriter produces more targeted edits, while the actor adapts with more elaborate reasoning. Additional analyzes are provided in Appendix I.3 and Appendix I.6.



(a) Strong ChartQA Actor: edits lower pass rate toward the sweet zone.



(b) Weak ZwZ Actor: edits raise pass rate toward the sweet zone.

Figure 4: **The EnvRewriter calibrates bi-directional difficulty.** On ChartQA, edits lower a strong Actor's pass rate; on ZwZ, edits raise a weak Actor's pass rate, moving both toward the sweet zone.

Table 7: **VICO remains robust under imperfect editing.** Training with 20% failed-edit images causes only a slight drop, showing that the co-evolution loop does not need a flawless image editor.

Baselines (↓)	**Mathematical Reasoning**						**Vision-Centric**				**Overall**
	MathVista	MathVision	ChartQA	MathVerse	WeMath	avg.	MMMU	MMVP	BLINK	avg.	avg.
Qwen3-VL-8B	67.7	31.5	76.9	50.6	72.0	59.8	55.8	77.7	68.9	67.5	63.6
Round1 (clean)	73.6	35.9	85.2	53.6	74.0	64.5	60.2	78.3	69.0	69.2	66.8
Round1 (noise-20%)	**72.9**	**35.5**	**84.8**	**52.2**	**72.7**	**63.6**	**58.4**	**78.0**	**68.9**	**68.5**	**66.0**

The iterative visual co-evolution loop drives the gains. We ablate whether the improvement comes from the co-evolution loop, the stronger offline data or RLVR-only training. First, we remove the loop and compare it against static offline augmentation. As shown in Table 5, stronger teachers improve the fixed corpus only modestly: Qwen3-VL-8B self-generation, GPT-5, and Nano Banana reach 78.64%, 80.10%, and 81.91% on ChartQA. All remain below 82%. In contrast, VICO-8B (Chart) reaches 89.12%, exceeding the strongest static method by +7.21%. Scaling the offline teacher from self-generation to Nano Banana gives only +3.27%, while adding the iterative loop gives +7.21%. Second, we keep the RLVR setup fixed but remove Rewriter-driven environment adaptation. Across all VICO variants, co-evolution outperforms Direct GRPO by +3.7% and Supervised GRPO by +4.8% on average. These results show that *the gains come from VICO's iterative, difficulty-aligned actor-environment co-evolution.*

VICO is robust against imperfect image editing. Automatic image editors may produce blank images, unchanged images, or image-label mismatches. We therefore stress-test VICO by corrupting 20% of the R1 training data. As shown in Table 7, the overall average performance drops by less than 1%. The noisy model still remains well above the baseline model. This shows that VICO does not require a flawless image editor: failed edits do not dominate training, and the remaining valid samples are sufficient for stable co-evolution. Additional details in Appendix H.4.

Image-generator capability does not replace co-evolution. Our editing strategy must control difficulty while keeping labels verifiable. Simple perturbations, such as noise, rotation, or blur, can lower the Actor's pass rate, but they mostly add perceptual noise and may shift samples away from the task distribution. Strong image generators produce better images, yet unconstrained generation can still cause answer drift or ambiguous labels. In contrast, VICO edits verifiable visual structures and preserves supervision through programmatic relabeling or answer-preserving editing. Thus, strong generators are useful backends, but not replacements for VICO: the core benefit is the actor-conditioned loop that selects, verifies, and re-aligns edits using rollout feedback. See Appendix J.

6 Conclusion

We present VICO, a co-evolution framework that reframes visual reasoning training as a dynamic, data-evolving co-evolution between an actor and an EnvRewriter, rather than a model-evolving process over a static corpus. By repurposing the rollout pass rate as a shared learning signal, VICO jointly trains an actor and an EnvRewriter that edits verifiable visual substructures toward the actor's learning frontier, producing difficulty-calibrated training samples without any additional human

annotation. Across nine multimodal benchmarks, VICO consistently outperforms strong RLVR, self-evolution, and co-evolution baselines, generalizes to out-of-distribution tasks unseen during training, and remains stable across multiple VLM backbones, exceeding the strongest self-evolution baseline by +4.3% on out-of-domain tasks while using up to 160× fewer labeled samples. Together, these results suggest that the bottleneck in visual reasoning post-training is *not the size of the static corpus but the fixed nature of the environment itself*, and that difficulty-aligned visual editing offers a scalable path forward in domains where high-quality human annotation is costly to obtain.

Acknowledgments and Disclosure of Funding

This research is sponsored by NSF 2442253, 2607580, NIH 1R21AG091260-01, R01EB039768, USDA NIFA 2027-67021-47662, Commonwealth Cyber Initiative, and generous gifts from Google, Amazon, Nvidia, and Cisco.

This research is supported by an Amazon Research Award, Spring 2026 Build on Trainium. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Amazon.

This research used the Delta system at the National Center for Supercomputing Applications [award OAC 2005572] through allocation [NAIRR240202] from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

[1] S. Alemohammad, J. Casco-Rodriguez, L. Luzi, A. I. Humayun, H. Babaei, D. LeJeune, A. Siahkoohi, and R. G. Baraniuk. Self-consuming generative models go mad, 2023.

[2] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, W. Ge, Z. Guo, Q. Huang, J. Huang, F. Huang, B. Hui, S. Jiang, Z. Li, M. Li, M. Li, K. Li, Z. Lin, J. Lin, X. Liu, J. Liu, C. Liu, Y. Liu, D. Liu, S. Liu, D. Lu, R. Luo, C. Lv, R. Men, L. Meng, X. Ren, X. Ren, S. Song, Y. Sun, J. Tang, J. Tu, J. Wan, P. Wang, P. Wang, Q. Wang, Y. Wang, T. Xie, Y. Xu, H. Xu, J. Xu, Z. Yang, M. Yang, J. Yang, A. Yang, B. Yu, F. Zhang, H. Zhang, X. Zhang, B. Zheng, H. Zhong, J. Zhou, F. Zhou, J. Zhou, Y. Zhu, and K. Zhu. Qwen3-vl technical report, 2025.

[3] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-vl technical report, 2025.

[4] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Rädle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Dollár, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer. Sam 3: Segment anything with concepts, 2026.

[5] L. Chen, J. Li, X. Dong, P. Zhang, C. He, J. Wang, F. Zhao, and D. Lin. Sharegpt4v: Improving large multi-modal models with better captions, 2023.

[6] L. Chen, X. Zhao, Z. Zeng, J. Huang, Y. Zhong, and L. Ma. Chart-r1: Chain-of-thought supervision and reinforcement for advanced chart reasoner, 2026.

[7] X. Chen, W. Hu, H. Li, J. Zhou, Z. Chen, M. Cao, Y. Zeng, K. Zhang, Y.-J. Yuan, J. Han, H. Xu, and X. Liang. C2-evo: Co-evolving multimodal data and model for self-improving reasoning, 2025.

[8] R. Dai, L. Song, H. Liu, Z. Liang, D. Yu, H. Mi, Z. Tu, R. Liu, T. Zheng, H. Zhu, and D. Yu. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models, 2025.

[9] Y. Deng, H. Bansal, F. Yin, N. Peng, W. Wang, and K.-W. Chang. Openvlthinker: Complex vision-language reasoning via iterative sft-rl cycles, 2025.

[10] Y. Deng, G. Chen, T. Gu, L. Kong, Y. Li, Z. Tang, and K. Zhang. Towards self-refinement of vision-language models with triangular consistency, 2025.

[11] A. Fortin, G. Vernade, K. Kampf, and A. Reshi. Introducing gemini 2.5 flash image, our state-of-the-art image model. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, Aug. 2025. Accessed: 2026-05-06.

[12] X. Fu, Y. Hu, B. Li, Y. Feng, H. Wang, X. Lin, D. Roth, N. A. Smith, W.-C. Ma, and R. Krishna. Blink: Multimodal large language models can see but not perceive, 2024.

[13] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Ding, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Chen, J. Yuan, J. Tu, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. You, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Zhou, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept 2025.

[14] J. Guo, L. Yang, P. Chen, Q. Xiao, Y. Wang, X. Juan, J. Qiu, K. Shen, and M. Wang. Genenv: Difficulty-aligned co-evolution between llm agents and environment simulators, 2025.

[15] J. He, J. Fang, F. Xiong, Z. Yao, F. Shen, H. Guo, J. Wang, and T.-S. Chua. Active zero: Self-evolving vision-language models through active environment exploration, 2026.

[16] Y. He, C. Huang, Z. Li, J. Huang, and Y. Yang. Visplay: Self-evolving vision-language models from images, 2025.

[17] Z. He, C. Zhang, Z. Wu, Z. Chen, Y. Zhan, Y. Li, Z. Zhang, X. Wang, and M. Qiu. Seeing is believing? mitigating ocr hallucinations in multimodal large language models, 2025.

[18] S. Huang, X. Qu, Y. Li, Y. Luo, Z. He, D. Liu, and Y. Cheng. Spotlight on token perception for multimodal reinforcement learning, 2025.

[19] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016.

[20] M. Kaledin, A. Golubev, and D. Belomestny. Variance reduction for policy-gradient methods via empirical variance minimization, 2022.

[21] N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith, Y. Wang, P. Dasigi, and H. Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025.

[22] Z. Li, H. Du, C. Huang, X. Wu, L. Yu, Y. He, J. Xie, X. Wu, Z. Liu, J. Zhang, and F. Liu. Mm-zero: Self-evolving multi-model vision language models from zero data, 2026.

[23] X. Liang, Z.-Z. Li, Y. Gong, Y. Wang, H. Zhang, Y. Shen, Y. N. Wu, and W. Chen. Sws: Self-aware weakness-driven problem synthesis in reinforcement learning for llm reasoning, 2025.

[24] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023.

[25] M. Lu, Y. Lu, Y. Zhuang, M. Mullins, Y. Xie, G. Xiao, C. Fleming, W. Shi, and X. Wang. Medvistagym: A scalable training environment for thinking with medical images via tool-integrated reinforcement learning, 2026.

[26] M. Lu, R. Xu, Y. Fang, W. Zhang, Y. Yu, G. Srivastava, Y. Zhuang, M. Elhoseiny, C. Fleming, C. Yang, Z. Tu, Y. Xie, G. Xiao, H. Wang, D. Jin, W. Shi, and X. Wang. Scaling agentic reinforcement learning for tool-integrated reasoning in vlms, 2025.

[27] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts, 2024.

[28] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022.

[29] F. Meng, L. Du, Z. Liu, Z. Zhou, Q. Lu, D. Fu, T. Han, B. Shi, W. Wang, J. He, K. Zhang, P. Luo, Y. Qiao, Q. Zhang, and W. Shao. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning, 2025.

[30] P. Movva and N. H. Marupaka. Enhancing scientific visual question answering through multimodal reasoning and ensemble modeling, 2025.

[31] OpenAI. Gpt-5 technical overview. https://openai.com, 2025. Accessed: 2026-05.

[32] R. Qiao, Q. Tan, G. Dong, M. Wu, C. Sun, X. Song, Z. GongQue, S. Lei, Z. Wei, M. Zhang, R. Qiao, Y. Zhang, X. Zong, Y. Xu, M. Diao, Z. Bao, C. Li, and H. Zhang. We-math: Does your large multimodal model achieve human-like mathematical reasoning?, 2024.

[33] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[34] G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297. ACM, Mar. 2025.

[35] J. Tong, J. Tang, H. Li, Y. Mou, M. Zhang, J. Zhao, Y. Wen, F. Song, J. Zhan, Y. Lu, C. Tao, Z. Guo, J. Yu, T. Cheng, Z. Xi, C. Jiang, Z. Yin, Y. Zheng, W. Ge, G. Chen, T. Gui, X. Qiu, Q. Zhang, and X. Huang. Game-rl: Synthesizing multimodal verifiable game data to boost vlms' general reasoning, 2025.

[36] H. Wang, Y. Yang, J. Hu, M. Zhu, and W. Chen. V-zero: Self-improving multimodal reasoning with zero annotation, 2026.

[37] K. Wang, J. Pan, W. Shi, Z. Lu, M. Zhan, and H. Li. Measuring multimodal mathematical reasoning with math-vision dataset, 2024.

[38] K. Wang, P. Zhang, Z. Wang, Y. Gao, L. Li, Q. Wang, H. Chen, C. Wan, Y. Lu, Z. Yang, L. Wang, R. Krishna, J. Wu, L. Fei-Fei, Y. Choi, and M. Li. Vagen: Reinforcing world model reasoning for multi-turn vlm agents, 2025.

[39] Q. Wang, B. Liu, T. Zhou, J. Shi, Y. Lin, Y. Chen, H. H. Li, K. Wan, and W. Zhao. Vision-zero: Scalable vlm self-improvement via strategic gamified self-play, 2025.

[40] Z. Wang, X. Guo, S. Stoica, H. Xu, H. Wang, H. Ha, X. Chen, Y. Chen, M. Yan, F. Huang, and H. Ji. Perception-aware policy optimization for multimodal reasoning, 2025.

[41] L. Wei, L. He, J. Lan, L. Dong, Y. Cai, S. Li, H. Zhu, W. Wang, L. Kong, Y. Wang, Z. Zhang, and W. Huang. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception, 2026.

[42] W. Wen, T. Zhi, K. Fan, Y. Li, X. Peng, Y. Zhang, Y. Liao, J. Li, and L. Zhang. Self-evolving vision-language models for image quality assessment via voting and ranking, 2026.

[43] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[44] Y. Xiao, E. Sun, T. Liu, and W. Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts, 2024.

[45] Y. Xie, Y. Ma, S. Lan, A. Yuille, J. Xiao, and C. Wei. Play to generalize: Learning to reason through game play, 2025.

[46] R. Xu, Y. Zhuang, Y. Zhong, Y. Yu, Z. Wang, X. Tang, H. Wu, M. D. Wang, P. Ruan, D. Yang, T. Wang, G. Xiao, X. Liu, C. Yang, Y. Xie, and W. Shi. Medagentgym: A scalable agentic training environment for code-centric reasoning in biomedical data science, 2025.

[47] Z. Xu, B. Qu, Y. Qi, S. Du, C. Xu, C. Yuan, and J. Guo. Chartmoe: Mixture of expert connector for advanced chart understanding, 2024.

[48] S. Yang, J. Li, X. Lai, B. Yu, H. Zhao, and J. Jia. Visionthink: Smart and efficient vision language model via reinforcement learning, 2025.

[49] Y. Yang, X. He, H. Pan, X. Jiang, Y. Deng, X. Yang, H. Lu, D. Yin, F. Rao, M. Zhu, B. Zhang, and W. Chen. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization, 2025.

[50] Y. Yang, Z. Zhang, Y. Hou, Z. Li, G. Liu, A. Payani, Y.-S. Ting, and L. Zheng. Effective training data synthesis for improving mllm chart understanding, 2025.

[51] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, and W. Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi, 2024.

[52] Y. Yue, Z. Chen, R. Lu, A. Zhao, Z. Wang, Y. Yue, S. Song, and G. Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025.

[53] H. Zhang, S. Ren, H. Yuan, J. Zhao, F. Li, S. Sun, Z. Liang, T. Yu, Q. Shen, and X. Cao. Mmvp: A multimodal mocap dataset with vision and pressure sensors, 2024.

[54] J. Zhang, S. Jin, C. Cheng, Y. Liu, Y. Lin, X. Zhang, Y. Zhang, F. Jiang, G. Yin, W. Lin, and R. Yan. Viper: Empowering the self-evolution of visual perception abilities in vision-language model, 2025.

[55] R. Zhang, D. Jiang, Y. Zhang, H. Lin, Z. Guo, P. Qiu, A. Zhou, P. Lu, K.-W. Chang, P. Gao, and H. Li. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems?, 2024.

[56] S. Zhang, Y. Zhang, J. Fu, L. Song, J. Bian, Y. Yang, and R. Wang. See less, see right: Bi-directional perceptual shaping for multimodal reasoning, 2025.

[57] H. Zhou, X. Li, R. Wang, M. Cheng, T. Zhou, and C.-J. Hsieh. R1-zero's "aha moment" in visual reasoning on a 2b non-sft model, 2025.

[58] J. Zhu, W. Wang, Z. Chen, Z. Liu, S. Ye, L. Gu, H. Tian, Y. Duan, W. Su, J. Shao, Z. Gao, E. Cui, X. Wang, Y. Cao, Y. Liu, X. Wei, H. Zhang, H. Wang, W. Xu, H. Li, J. Wang, N. Deng, S. Li, Y. He, T. Jiang, J. Luo, Y. Wang, C. He, B. Shi, X. Zhang, W. Shao, J. He, Y. Xiong, W. Qu, P. Sun, P. Jiao, H. Lv, L. Wu, K. Zhang, H. Deng, J. Ge, K. Chen, L. Wang, M. Dou, L. Lu, X. Zhu, T. Lu, D. Lin, Y. Qiao, J. Dai, and W. Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025.

A Limitations and Impact Statement

Limitations Although VICO shows strong empirical gains across multimodal reasoning benchmarks, several limitations remain. First, VICO addresses structured or constrained verifiable visual editing for RLVR. It requires an editable representation whose answers can be recomputed, or sufficiently reliable grounding to constrain edits that preserve an existing answer. CLEVR and ChartQA provide scene graphs and chart tables; the natural-image VQA arm uses protected regions and does not perform unrestricted semantic scene rewriting. Structure can sometimes be recovered with a tool rather than supplied by the dataset (Appendix C.8), but domains without recoverable structure or reliable answer grounding remain outside the current scope.

Second, VICO is human-QA-label efficient under pre-existing or recoverable verifiable structure. This claim concerns additional QA annotation during co-evolution; it does not remove the cost of seed annotations, metadata, renderers, image editors, GPT-5 edit demonstrations for EnvRewriter initialization, or multi-round training. Appendix G.3 separates these resources. Reusing Actor rollouts for difficulty scoring avoids a second set of rollouts for that statistic, but does not eliminate EnvRewriter training or editing costs.

Third, our evaluation focuses on visual reasoning tasks with verifiable answers. Open-ended generation, dialog, and real-time interactive environments are outside the current scope. Future work can study adaptive difficulty targets, broader visual domains, and tighter integration with more general-purpose image editors.

Fourth, passing an execution or structural check does not always establish semantic label validity. The manual audit in Appendix C.7 detects drift in $11/200$ retained ChartQA Tier 2 samples and none in 200 retained ZwZ samples. These finite audits characterize the evaluated samples rather than guaranteeing error-free editing. Similarly, the analysis in Appendix E motivates the curriculum design under simplified assumptions; it is not a convergence or optimality guarantee for the complete VLM training pipeline.

Impact Statement VICO aims to reduce the reliance on large-scale human annotation for VLM post-training by turning existing visual data into adaptive, label-valid training samples. This may make visual reasoning training more scalable and accessible, especially in domains where high-quality annotations are expensive. At the same time, automatic image editing can introduce artifacts, biases, or label drift if used without verification. Our framework mitigates these risks through deterministic relabeling, answer-preserving constraints, and filtering, but careful validation remains necessary before applying similar systems to high-stakes domains.

B Implementation Details

B.1 VICO: Algorithm Details

VICO trains an Actor and an EnvRewriter in an iterative actor–environment loop. The Actor learns from verifiable rollout rewards, while the EnvRewriter learns from the Actor's empirical pass rate on its edited samples. The two models share a common pool of rollouts: each rollout simultaneously yields a task-level reward used to update the Actor and a difficulty signal used to update the EnvRewriter, enabling co-evolution without doubling the inference cost. The complete training loop is summarized in Algorithm 1 and described below.

Inputs and initialization The pipeline starts from a seed visual dataset $\mathcal{D}_{\text{seed}} = \{(x, q, a, \sigma)\}$, where x is the image, q is the question, a is the answer, and σ is a verifiable substructure when available, such as a CLEVR scene graph or a ChartQA data table. We initialize the Actor $\pi_a^{(0)}$ and the EnvRewriter $\pi_R^{(0)}$ from the same pretrained VLM family. The EnvRewriter is first SFT-initialized to produce executable edit commands. Five hyperparameters control the loop: the number of rounds T, the rollouts per sample K, the target pass rate α (we use $\alpha = 0.5$), the difficulty filter threshold $k_{\min}$ that discards edits whose pass rate is too far from α, and the RWR temperature λ that controls how sharply the EnvRewriter is biased toward high-reward edits. Two online pools, the Actor pool $\mathcal{D}_a$ and the EnvRewriter pool $\mathcal{D}_R$, are initialized empty and grow throughout training.

Algorithm 1 VICO jointly updates the Actor and EnvRewriter from shared rollouts. Each round generates edited tasks, evaluates Actor rollouts, updates the Actor with task rewards, and updates the EnvRewriter with difficulty-calibration rewards.

Require: Actor π_a, EnvRewriter π_R, seed dataset $\mathcal{D}_{\text{seed}} = \{(x_i, q_i, a_i, \sigma_i)\}$, number of rounds T, rollouts per sample K, target pass rate α, difficulty filter threshold $k_{\min}$, RWR temperature λ
Ensure: Trained actor π_a, EnvRewriter π_R
Initialize actor pool $\mathcal{D}_a \leftarrow \varnothing$, EnvRewriter pool $\mathcal{D}_R \leftarrow \varnothing$
for round $t = 1, \ldots, T$ **do**
  Stage 1: Online Generation & Interaction
  Sample one edit command $u_i \sim \pi_R(\cdot \mid x_i, \sigma_i)$ per seed; render $x'_i = \mathcal{R}(u_i(\sigma_i))$
  Build $\mathcal{T}_t$ from edited triples passing execution and domain-specific validity checks
  for each edited triple $(x'_i, q'_i, a'_i) \in \mathcal{T}_t$ **do**
    Sample K rollouts $\{\hat{a}_{i,k}\}_{k=1}^{K}$ from $\pi_a(\cdot \mid x'_i, q'_i)$
    Compute actor reward $R_{\text{actor}}(\hat{a}_{i,k}, a'_i)$ for each rollout
    Compute pass rate $\hat{p}_i \leftarrow k_i / K$
    Compute EnvRewriter reward $R_{\text{rewriter}}(\hat{p}_i)$
  end for
  Stage 2: Joint Update
  Update π_a via GRPO on $\mathcal{T}_t$ to maximize $\mathbb{E}[R_{\text{actor}}]$
  Filter: $\tilde{\mathcal{T}}_t \leftarrow \{(x_i, \sigma_i, u_i) : |\hat{p}_i - \alpha| \le k_{\min}\}$
  Build weighted SFT set: $\tilde{\mathcal{D}}_R^t \leftarrow \{(x_i, \sigma_i, u_i, w_i) : w_i \propto \exp(\lambda R_{\text{rewriter}}(\hat{p}_i))\}$
  Update π_R via RWR on $\tilde{\mathcal{D}}_R^t$ to maximize $\mathbb{E}[R_{\text{rewriter}}]$
  Stage 3: Aggregation
  $\mathcal{D}_a \leftarrow \mathcal{D}_a \cup \text{Valid}(\mathcal{T}_t)$
  $\mathcal{D}_R \leftarrow \mathcal{D}_R \cup \tilde{\mathcal{D}}_R^t$
end for

Stage 1: Online generation and interaction At the start of each round t, the EnvRewriter proposes a structured edit command $u_i \sim \pi_R(\cdot \mid x_i, \sigma_i)$ over the verifiable substructure σ_i for every seed instance. A domain renderer or editor produces the edited image $x'_i = \mathcal{R}(u_i(\sigma_i))$, and the answer a'_i is either computed programmatically from the edited structure or preserved by construction under the domain-specific validity rules in Appendix C. Retained samples satisfy the implemented checks; residual semantic errors in the preservation arms are quantified in Appendix C.7. For each edited triple (x'_i, q'_i, a'_i), the Actor samples K responses, and each response receives the rule-based Actor reward defined in Appendix B.4. We then compute the empirical pass rate $\hat{p}_i = k_i/K$, where k_i is the number of correct rollouts, and map it to the difficulty-calibration reward R_{rewriter}, which peaks when $\hat{p}_i$ matches the target α.

Edit sampling and rollout reuse. Each seed produces one EnvRewriter candidate per round. We do not sample a candidate pool and select its best-scoring edit. A rendered candidate is checked for execution and domain-specific validity before Actor evaluation; the difficulty filter subsequently determines its eligibility for the EnvRewriter update. For each valid candidate, the Actor samples $K = 8$ responses with temperature 0.7 and top-$p = 1.0$. The same responses supply the Actor rewards and the empirical pass rate used to score the edit, so difficulty assignment does not require a separate Actor inference pass. This reuse does not remove the cost of generating or optimizing edit commands.

Stage 2: Joint update The two models are updated from the same batch of rollouts, but with different reward signals. Valid edited triples (x'_t, q'_t, a'_t) are used to update the Actor π_a via GRPO to maximize $\mathbb{E}[R_{\text{actor}}]$, leveraging the within-group advantage normalization that makes GRPO robust to scale differences across edits. The EnvRewriter π_R is updated via Reward-Weighted Regression (RWR): we first apply a difficulty filter, retaining only edits whose pass rate falls within $[\alpha - k_{\min}, \alpha + k_{\min}]$, and then construct a weighted SFT set in which each retained edit transition is weighted by $\exp(\lambda R_{\text{rewriter}})$. Fine-tuning π_R on this set biases it toward edits that land close to the target band. We additionally apply a KL penalty to the initial EnvRewriter and cap the per-step KL to prevent the editor from drifting into degenerate or off-distribution edit distributions.

Stage 3: Aggregation After each round, valid edited triples (those that render successfully and pass label-consistency checks) are appended to the Actor pool $\mathcal{D}_a$, while the weighted edit transitions are appended to the EnvRewriter pool $\mathcal{D}_R$. Both pools persist across rounds, so the Actor sees an increasingly diverse set of edited tasks, and the EnvRewriter learns from an increasingly informative

history of difficulty signals. The updated Actor and EnvRewriter then become the starting point for the next round.

Why this differs from static RLVR? The key distinction from static RLVR is that the empirical pass rate $\hat{p}$ is not used only as a filtering statistic. It is also reused as an actor-relative supervision signal for the EnvRewriter, allowing the visual environment to track the Actor's current capability. Crucially, the loop requires no new human-labeled QA: the seed representation σ_i provides the editable interface, and labels for edited samples are either recomputed or preserved through explicit validity constraints.

B.2 Actor GRPO Training Details

For the Actor π_a, we use GRPO with rule-based rewards. For each edited prompt, we sample multiple on-policy rollouts and score them with exact-match answer verification plus a format check. Unless otherwise stated, rollout sampling uses temperature 0.7, top-$p = 1.0$, and a maximum response length of $3{,}824$ tokens, matching the zero-shot evaluation setting in the main text.

In the ChartQA multi-round runs, each round trains on valid EnvRewriter-generated modified training images, together with a fixed regularization set of $2{,}000$ original ChartQA samples. The resulting Actor checkpoint initializes the next round.

B.3 EnvRewriter SFT and RWR Details

The EnvRewriter is trained in two stages. First, we initialize it with format-aware SFT on verified edit demonstrations generated by GPT-5. This stage teaches the model to produce structured, executable edit commands, use the available edit operations, and propose edits that adjust task difficulty while preserving label verifiability. Second, during co-evolution, we update the learned EnvRewriter using reward-weighted regression over valid edit transitions collected through interaction with the Actor.

For each executed edit, the current Actor produces K rollouts on the resulting sample. The empirical pass rate $\hat{p}$ is used to compute the edit's RWR weight according to the bell-shaped difficulty objective in Appendix B.4. These weights are computed from the outcome of each edit turn. Invalid or unexecutable edits, as well as edits outside the RWR difficulty band, are excluded from the RWR target set. This encourages the EnvRewriter to concentrate its probability mass on valid edits near the Actor's learnable zone. Actor training data are constructed separately: samples that satisfy execution and label-validity requirements may be retained even when their pass rates fall outside the RWR difficulty band.

GPT-5 as a strategic trajectory generator. We use GPT-5 to generate editing demonstrations for training the EnvRewriter. Given a seed sample, its editable structure, the current Actor's empirical pass rate, the target difficulty band, and the domain-specific edit schema, GPT-5 proposes a structured edit command intended to adjust the task's difficulty. The command is executed by the environment, and the resulting sample is evaluated by the Actor. The execution outcome and Actor feedback can then be included in the context for subsequent edits, producing actor-conditioned editing trajectories. GPT-5 provides the edit proposals, but it does not serve as the authority for either ground-truth labels or training rewards. Labels are obtained through domain-specific recomputation or checked for answer preservation, depending on the edit setting, while difficulty is measured using Actor rollouts. We construct the SFT dataset from executable, verified demonstrations whose edits move the Actor's pass rate closer to the target pass rate. These demonstrations supervise both the edit-command format and the choice of editing strategies.

EnvRewriter interface. The EnvRewriter is an LVLM policy initialized from the same model family as the Actor and SFT-initialized to produce executable edit commands. Its input contains the image, question, answer, editable structure σ, current Actor pass rate $\hat{p}$, target pass rate α, and the domain-specific tool schema. Its output is a structured tool call with fields `tool`, `params`, and `reasoning`. For example, in ChartQA, the EnvRewriter may output a `chart_rerender` command that edits the underlying CSV table and re-renders the chart. The command is executed by the domain renderer or image editor, and the resulting sample is kept only if it passes the validity checks in Table 1. Full prompts and tool schemas are provided in Appendix F.4.

B.4 Reward Functions and Difficulty Filtering

Actor Reward. The Actor reward is defined as:

$$R_{\text{actor}}(q', o) = \begin{cases} +1, & \text{if } \text{FormatCorrect}(o) \wedge \text{AnswerCorrect}(o), \\ -1, & \text{otherwise.} \end{cases} \tag{B.1}$$

EnvRewriter Reward. For each executed edit, the empirical pass rate on the resulting sample is $\hat{p} = k/K$, where k is the number of correct Actor rollouts under R_{actor}. The EnvRewriter reward is:

$$R_{\text{rewriter}}(\hat{p}) = \exp\left(-\beta(\hat{p} - \alpha)^2\right), \tag{B.2}$$

where $\beta > 0$ controls sharpness. We use $\alpha = 0.5$, $\beta = 4.0$, and $k_{\min} = 0.15$ for the difficulty filter.

Difficulty filter. The reward encourages edits whose resulting samples have pass rates near the Actor's learning frontier. In practice, we keep a sweet-zone band around $\alpha = 0.5$ and discard invalid or poorly calibrated edits from EnvRewriter updates. This prevents the EnvRewriter from overfitting to samples that are already solved or impossible for the current Actor.

B.5 Hyperparameters

Table 8: **VICO uses one compact training recipe.** Unless stated otherwise, the same core hyperparameters are used across visual domains.

Component	Value
Actor optimizer	AdamW
Actor update	GRPO
Actor learning rate	1×10^{-6}
Rollouts per sample	$K = 8$ for pass-rate estimation
Sampling temperature	0.7
Top-p	1.0
Maximum response length	3,824 tokens
EnvRewriter update	Reward-weighted LoRA SFT / RWR
Difficulty target	$\alpha = 0.5$
Reward sharpness	$\beta = 4.0$
Sweet-zone band	$[0.35, 0.65]$ unless otherwise stated
Training rounds	Up to 5 rounds
Hardware	Up to 8×A100-80GB GPUs
Training framework	verl-based distributed RL training

RL training parameters. We optimize the actor with reinforcement learning to improve multi-step visual reasoning.

B.6 Compute and Hardware

All experiments use distributed training on up to 8×A100-80GB GPUs. We use a VERL-based implementation for GRPO training and parallel rollout generation. A detailed cost and annotation-efficiency comparison is provided in Appendix G.3.

C Data Sources and Verifiable Visual Editing

This appendix describes how VICO produces label-valid training samples across three visual domains. The key point is that different domains provide different forms of verifiability.

For CLEVR, VICO edits an explicit scene graph and recomputes the answer programmatically by executing the original question program. For ChartQA, VICO edits the underlying CSV table and uses a two-tier verification rule: Tier 1 recovers the new answer by entity-anchored lookup in the edited table, while Tier 2 keeps CSV values fixed and preserves the original answer by construction. For

Table 9: **RWR trains the EnvRewriter with pass-rate-weighted edit transitions.** We train LoRA on Qwen3VL-8B-Instruct using LlamaFactory. Each retained edit transition is duplicated $\lfloor 5 \cdot w(\hat{p}) \rfloor$ times based on the Actor pass rate $\hat{p}$ on the sample produced by that edit turn.

Model & LoRA		*Optimization*	
base model	Qwen3VL-8B-Instruct	learning rate	5.0×10^{-6}
template	qwen3_vl	scheduler	cosine
finetuning type	LoRA	warmup ratio	0.1
LoRA rank	64	total epochs	10
LoRA alpha	128	per-device batch	2
LoRA target	all	grad. accum. steps	8
precision	bf16	effective batch	64
Eval & Checkpointing		*RWR Data Construction*	
eval strategy	steps	RWR weight α	0.5
eval/save steps	50 / 50	RWR weight β	4.0
save total limit	3	max sample repeats	5
load best model	true	weight formula	$\exp\big(-\beta(\hat{p}-\alpha)^2\big)$
best metric	eval_loss	number of GPUs	4

Table 10: **GRPO trains the Actor with rollout-level verifiable rewards.** We train Qwen3VL-8B-Instruct with verl and a vLLM rollout backend. We disable KL penalty in reward shaping and loss to avoid gradient interference, following [33].

Model & Reward		*Rollout & Batch*	
base model	Qwen3VL-8B-Instruct	rollouts per prompt, N	8
algorithm	GRPO	train batch size	32
reward manager	binary	PPO mini-batch	8
KL in reward	False	PPO micro-batch/GPU	1
KL loss coeff.	0	log-prob micro-batch/GPU	1
entropy coeff.	0	rollout backend	vLLM
		tensor model parallel	1
		sampling temperature	0.7
Optimization		*Sequence Limits & Schedule*	
actor learning rate	1.0×10^{-6}	max prompt length	1,048
scheduler	cosine	max response length	3,824
warmup ratio	0.05	max model length	6,896
total epochs	3	max batched tokens	16,392
precision	bf16	truncation	left
grad. checkpointing	enabled	filter overlong prompts	True
FSDP strategy	FSDP2	save / val freq	50 / 20
sequence parallel size	1	validate before training	True
number of GPUs	8		

natural-image VQA, where no complete symbolic state is available, VICO uses answer-preserving regional editing: the question and answer remain fixed, while the EnvRewriter edits only non-target visual regions.

Thus, VICO uses two label-validity mechanisms: deterministic recomputation when the answer can be derived from the edited structure, and constrained answer preservation when the answer-bearing structure must remain fixed.

C.1 Seed Data Sources

VICO uses a small seed set to initialize each visual reasoning domain. For compositional reasoning, we sample CLEVR images with their scene graphs and question programs. For chart reasoning, we sample ChartQA training images together with their underlying data tables. For natural-image VQA, we sample ZwZ images, which provide answer-relevant region annotations. We use these regions to identify target objects and preserve the original question-answer pair during editing.

C.2 Overview of the Three Editing Arms

Table 11 summarizes the three editing arms. CLEVR supports full deterministic answer recomputation from the edited scene graph. ChartQA uses a two-tier table rule: Tier 1 updates the answer by entity-anchored lookup in the edited CSV table, while Tier 2 preserves the answer under constrained table edits. The natural-image VQA arm follows a different rule: it keeps the original question and answer unchanged and only edits regions outside the answer-relevant target objects.

Table 11: **VICO uses three editing arms for verifiable visual variation.** CLEVR recomputes answers from scene graphs, ChartQA uses a two-tier table rule, and ZwZ preserves answers by protecting target objects.

Arm	Editing interface	Answer rule
CLEVR	Scene graph edits over objects, attributes, and relations; rendered with Blender.	Recompute a' by re-executing the question program.
ChartQA	CSV/table edits rendered with matplotlib.	Tier 1: entity-anchored lookup; Tier 2: keep $a' = a$ under constrained edits.
ZwZ	SAM3 masks with protected target objects; edit only non-target regions.	Keep $q' = q$ and $a' = a$ under target-object protection.

C.3 CLEVR: Scene-Graph Editing

For CLEVR, the editable structure σ is the scene graph paired with each image. Each scene graph specifies a set of objects, with each object carrying attributes such as `shape`, `color`, `material`, and `size`, as well as pairwise spatial relations such as `left-of`, `right-of`, `front`, and `behind`. Questions are generated from question-program templates over the scene graph, so the same program can be re-executed on the edited graph to produce a'.

Edit operations. The EnvRewriter outputs an edit command u in one of four categories: (i) **attribute change**, which modifies the `color`, `material`, or `size` of a target object; (ii) **category swap**, which changes an object's `shape` among `cube`, `sphere`, and `cylinder`; (iii) **entity add/remove**, which inserts or deletes objects while preserving renderability; and (iv) **relation modification**, which repositions objects to alter spatial relations. Each operation acts on the scene graph rather than on pixels, so $u(\sigma)$ remains a fully specified renderable scene.

Renderer. We use Blender with the original CLEVR rendering script to produce $x' = \mathcal{R}(u(\sigma))$. Camera parameters, lighting, and material shaders are kept consistent with the original rendering pipeline to avoid stylistic drift.

Answer derivation. We retain the original question program and re-execute it on $u(\sigma)$ to obtain a'. For example, if the original program is `count(filter(color=red, shape=cube))` and the edit changes one red cube into a blue cube, the new answer decreases by one. Because program execution on a discrete scene graph is exact, CLEVR samples are label-valid by construction.

Failure mode. An edit is discarded if it makes the question ambiguous or non-executable. For example, if an edit creates multiple possible referents for a question that assumes a unique object, the sample is removed before training.

C.4 ChartQA: Table-Level Editing and Re-rendering

For chart reasoning, the editable structure σ is the CSV table underlying each chart. The EnvRewriter edits the table-level representation rather than chart pixels, and matplotlib re-renders the edited table into a new chart image. This allows VICO to adjust visual and numerical difficulty while keeping label validity controlled by the table.

Tier construction. For each ChartQA seed sample, we first apply an automatic label-to-cell matching heuristic. We scan the CSV table and match value cells against the original answer. If exactly one cell matches, the sample is assigned to Tier 1. Let $(r^\star, j^\star)$ denote the matched answer cell, and let $e^\star$ denote the row entity at $r^\star$ in the original table. If no cell matches, or multiple cells

match, the sample is assigned to Tier 2. Tier 2 often includes derived-answer cases such as ratios, percentages, or differences, or ambiguous duplicate values.

Edit operations. For Tier 1, the EnvRewriter may modify table values, add near-match distractor rows, increase chart density, or sharpen contrasts. After editing, the answer is recovered by entity-anchored lookup in the edited table:

$$a' = \text{table}'[\text{entity} = e^\star, j^\star].$$

This lookup avoids relying on the original row index after row insertion or reordering. For Tier 2, the EnvRewriter keeps CSV values fixed and only varies rendering style or other answer-preserving visual factors. The label is kept as $a' = a$ under this preservation rule. Its semantic reliability is assessed separately in Appendix C.7.

Renderer. We re-render charts with matplotlib using the same chart type and style configuration as the source chart whenever possible, including chart type, color palette, axis labels, fonts, and gridlines.

Failure mode. We discard edits that violate the table schema, remove or duplicate the Tier 1 anchor entity $e^\star$, remove the target column $j^\star$, produce no numeric columns, fail rendering, or break the Tier 2 invariance constraint by changing CSV values. Numerical edits are bounded to keep the rendered chart visually plausible.

C.5 ZwZ: Answer-Preserving Natural-Image Editing

For natural-image VQA, VICO does not attempt to recompute a new answer from a full symbolic scene representation. Instead, it preserves the original question-answer pair. Given a seed sample (x, q, a), the edited sample is (x', q, a), where $q' = q$ and $a' = a$.

The key assumption is that the answer is grounded in a set of answer-relevant target objects. ZwZ provides answer-relevant region annotations, and we use SAM3 to decompose the image into object-level masks. Masks that overlap the answer-relevant region are treated as **target objects**. These target objects form the protected region σ^+. All other object masks and background regions form the editable complement σ^-.

Edit operations. The EnvRewriter receives the image, question, answer, object masks, protected target objects, and current Actor pass-rate signal. It then proposes edits restricted to σ^-. Typical edits include adding distractor objects outside the target region, modifying background clutter, changing the appearance of non-target objects, or introducing visually plausible distractors that do not alter the answer-relevant target objects. The EnvRewriter is not allowed to edit, occlude, remove, or semantically change the protected target objects.

Rendering. The original image, the protected target-object mask, and the edit command are passed to an external image editor. The editor modifies only the editable complement and returns the edited image x'. Unlike CLEVR or ChartQA, this step is not treated as deterministic symbolic rendering. It is an answer-preserving image-editing step under a protected-region constraint.

Answer rule. For ZwZ, the answer is not recomputed. We keep the original question and answer:

$$q' = q, \qquad a' = a.$$

The Actor reward therefore compares the model's prediction against the original answer a. This distinguishes natural-image VQA from the structured arms, where a' is programmatically recomputed.

Failure mode. An edit is discarded if it touches the protected target objects, changes the visual evidence needed to answer the question, or introduces new content that would make the original answer ambiguous. For example, if the question depends on counting target objects, adding another object of the same relevant category is rejected. If the question depends on a spatial relation involving a target object, edits that change that relation are rejected. This conservative filtering keeps the natural-image VQA arm answer-preserving rather than relabeled.

Table 12: **Execution validity and semantic label validity are distinct.** Retention is the reported valid-edit retention in the generation audit. Programmatic checks cover retained structured edits; manual checks examine 200 retained samples per answer-preservation domain. A zero detected rate is specific to the stated check and does not establish zero semantic error.

Domain	Validity mechanism	Retention	Audit	Detected drift
CLEVR	Program re-execution	58.6%	Programmatic checks	0% detected
ChartQA Tier 1	Entity-anchored lookup	97.9%	Programmatic checks	0% detected
ChartQA Tier 2	Answer preservation	100%	200 manual samples	5.5% (11/200)
ZwZ	Protected-region editing	99.1%	200 manual samples	0% (0/200)

C.6 Filtering Invalid or Ambiguous Edits

Across all editing arms, we discard edits that fail label-validity or execution checks.

First, the edited sample must be executable by the corresponding renderer or image editor. For CLEVR, the scene graph must render successfully in Blender. For ChartQA, the edited CSV must satisfy the table schema and render successfully with matplotlib. For ZwZ, the image editor must preserve the protected target-object region.

Second, the edited sample must preserve a valid answer. For CLEVR, the question program must be executable on the edited scene graph to compute a'. For ChartQA Tier 1, the anchor entity $e^\star$ and target column $j^\star$ must remain valid and addressable in the edited table so that $a' = \mathrm{table}'[\mathrm{entity} = e^\star, j^\star]$ can be recovered by lookup. For ChartQA Tier 2, the edit must satisfy the invariance constraint: CSV values remain fixed and the original answer is kept as $a' = a$. For ZwZ, the original answer is retained under the protected-region constraint; the independent audit in Appendix C.7 checks whether the resulting image still supports it.

Third, the edited sample must provide useful difficulty for the current Actor. We measure this by the empirical pass rate $\hat{p}$ after Actor rollouts. Invalid, ambiguous, unexecutable, or out-of-band edits are removed from the EnvRewriter update set.

C.7 Label-Validity Audit and Semantic Failure Modes

We distinguish successful execution, structural consistency, and semantic support for the question. The first two can be checked automatically in the structured arms, whereas preserving an answer string or a target mask does not by itself establish the third. We therefore combine an automatic audit of generation logs with manual inspection of 200 retained ChartQA Tier 2 samples and 200 retained ZwZ samples (Table 12).

Structured edits. CLEVR retains 58.6% of proposed edits, and ChartQA Tier 1 retains 97.9%. No label inconsistency is detected by the respective programmatic checks among the retained samples. These checks establish agreement with the executable question program or the selected table anchor; they are not an independent semantic audit of the natural-language question. In particular, Tier 1 relies on the correctness of its question-to-cell interpretation. CLEVR's lower retention reflects conservative validation of scene edits. A changed answer is permitted when the question remains executable and the label is recomputed; a label-changing edit is not intrinsically invalid.

Answer-preservation edits. Despite 100% automatic retention, ChartQA Tier 2 exhibits edit-induced semantic drift in $11/200$ manually inspected samples (5.5%): seven become ambiguous and four lose a cue needed to answer the question. Thus, keeping CSV values unchanged is insufficient when a rendering change alters the visual evidence referenced by the question. ZwZ retains 99.1% of proposed edits and exhibits no edit-induced drift in the 200 inspected retained samples. The latter observation does not establish a zero population error rate. Source-data inconsistencies already present before editing are distinguished from edit-induced drift, because these measure different failure mechanisms.

Implications for training. The artificial corruption study in Appendix H.4 shows some tolerance to failed edits, but its blank-image and mismatched-label corruptions differ from the semantic errors found here. It therefore does not certify robustness to every form of label drift.

Table 13: **Visual co-evolution also benefits from tool-recovered chart structure.** All rows are evaluated on real ChartQA test images. The provided-structure result is the main R4 reference; the recovered-structure and text-only experiments are reported through two iterations.

Setting	Structure	Label source	ChartQA
Base	None	—	76.9
VICO, provided structure (R4)	Dataset CSV	Recomputed / preserved	89.1
VICO, recovered (Iter. 1)	ChartMoE table	Computed from table	83.9
VICO, recovered (Iter. 2)	ChartMoE table	Computed from table	86.3
Text-only evolution (Iter. 1)	None	Majority vote	77.4
Text-only evolution (Iter. 2)	None	Majority vote	78.1

Additional semantic validation. A complementary safeguard is a VLM-based semantic-consistency check on the edited image, question, and retained answer before admitting a sample to training. Such a check should test whether the question remains unambiguous and whether the image still supports the answer, rather than only whether the answer string or protected pixels are unchanged. The manual audit above does not measure this filter's effectiveness; quantifying its false acceptances, false rejections, and residual label drift requires a separate evaluation.

C.8 Recovering Editable Structure from Chart Images

To test whether the visual editing interface must be supplied by the dataset, we use ChartMoE [47] to recover a table from a chart image. The EnvRewriter edits the recovered table, and the renderer generates a new chart with randomized visual style parameters. Labels are computed from the table used to render the new image, rather than obtained by asking a VLM to answer the rendered chart. This aligns the synthetic image and its underlying data, but still requires a valid question-to-table mapping; table recovery alone does not certify question semantics or recover every detail of the original image.

On real ChartQA test images, recovered-structure VICO reaches 83.9% after one iteration and 86.3% after two. At the second iteration it exceeds the text-only evolution result (78.1%) by 8.2 percentage points. Relative to the 76.9% base reference, its 9.4-point gain is approximately 77% of the 12.2-point gain of the main provided-structure run. The latter is an R4 reference rather than a matched two-iteration comparison. These results support using recovered structure in the chart arm; they do not establish applicability to arbitrary images or replace the need for reliable grounding in natural-image VQA.

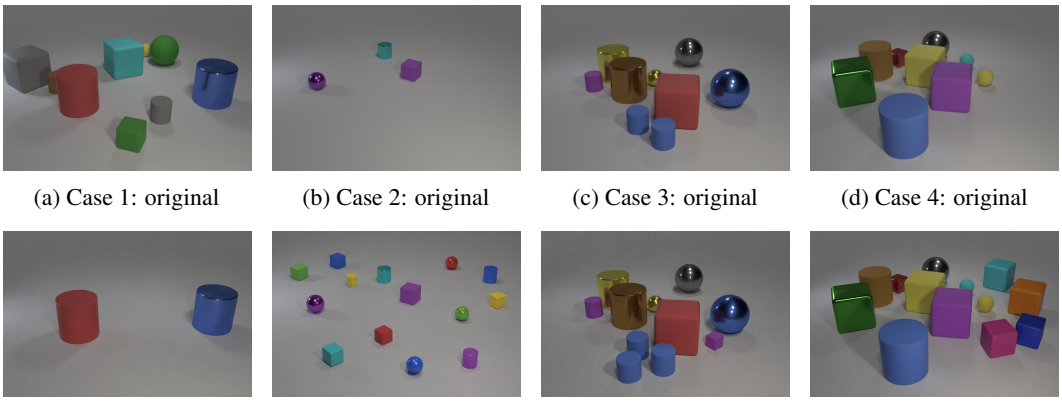
C.9 Why Constrain Edits Instead of Repeatedly Pseudo-Labeling Images?

An alternative is to generate visual content freely and ask a stronger model or a voting ensemble to label it. In RLVR, incorrect labels directly affect the reward assigned to Actor responses. VisPlay [16] also reports estimated pseudo-label accuracies of 72%, 65%, and 61% on successive generated question sets, using ChatGLM-Flash for the quality assessment. These estimates concern a different pipeline and should not be directly compared with our manual audit rates.

VICO reduces repeated QA labeling by restricting the editable structure and recomputing or retaining the answer under domain-specific rules. This trades editing freedom for more explicit control over supervision. It does not eliminate semantic errors in the preservation arms, as Appendix C.7 demonstrates. V-Zero [36], for example, constrains its generated training questions to four-option multiple choice and uses majority voting. Our restriction is placed on the visual edit, allowing numerical and short free-form answers when the domain interface can verify them; it is not a claim that voting can never support such answers.

C.10 Dataset and Edited Sample Visualizations

Figures 5, 6, and 7 show representative visual examples for the three editing arms. For CLEVR, edits are applied to a scene graph and the answer a' is recomputed by executing the original question program. For ChartQA, Tier 1 examples recover a' by entity-anchored lookup in the edited CSV table, while Tier 2 examples preserve $a' = a$ through constrained table edits. For ZwZ, the original question and answer are retained, while answer-relevant target objects are protected and only non-target



(a) Case 1: original (b) Case 2: original (c) Case 3: original (d) Case 4: original

(e) Case 1: easier (f) Case 2: harder (g) Case 3: harder (h) Case 4: harder

Figure 5: **CLEVR edits change difficulty while preserving verifiable labels.** Each column shows one case before and after editing. The EnvRewriter modifies the scene graph, re-renders the image, and recomputes a' by executing the original question program on the edited graph.

regions are edited. Together, these examples show the two label-validity mechanisms used by VICO: deterministic recomputation when an answer can be derived from the edited structure, and constrained answer preservation when the answer-bearing structure must remain fixed.

D Image Editing Case Studies

D.1 CLEVR Scene-Graph Editing Examples

Figure 5 shows representative CLEVR edits. Each example starts from a seed scene and applies a structured edit to object attributes, object categories, object counts, or spatial relations. Because the edit is applied to the scene graph rather than directly to pixels, the modified scene remains renderable, and the answer can be recomputed by program execution. These examples illustrate how VICO produces label-valid visual perturbations instead of relying on model-based relabeling.

D.2 ChartQA Table-Level Editing Examples

We present the ChartQA edits shown in Figure 6. Each seed sample is associated with an underlying data table table, a question q, and an original answer a. Conditioned on the current Actor pass rate $\hat{p}$, the EnvRewriter generates a table-level edit aimed at the desired difficulty. To make the sample harder, it may add near-match rows, distractor series, or more confusable values; to make it easier, it may remove distractors or sharpen relevant contrasts. The edit is applied to table to produce table$'$, which is then re-rendered into a new chart.

We use the same two-tier verification rule as in Table 1. For Tier 1 samples, the answer maps to a unique CSV cell $(r^\star, j^\star)$. Let $e^\star$ denote the row entity at $r^\star$ in the original table. After editing, the new answer is recovered by entity-anchored lookup:

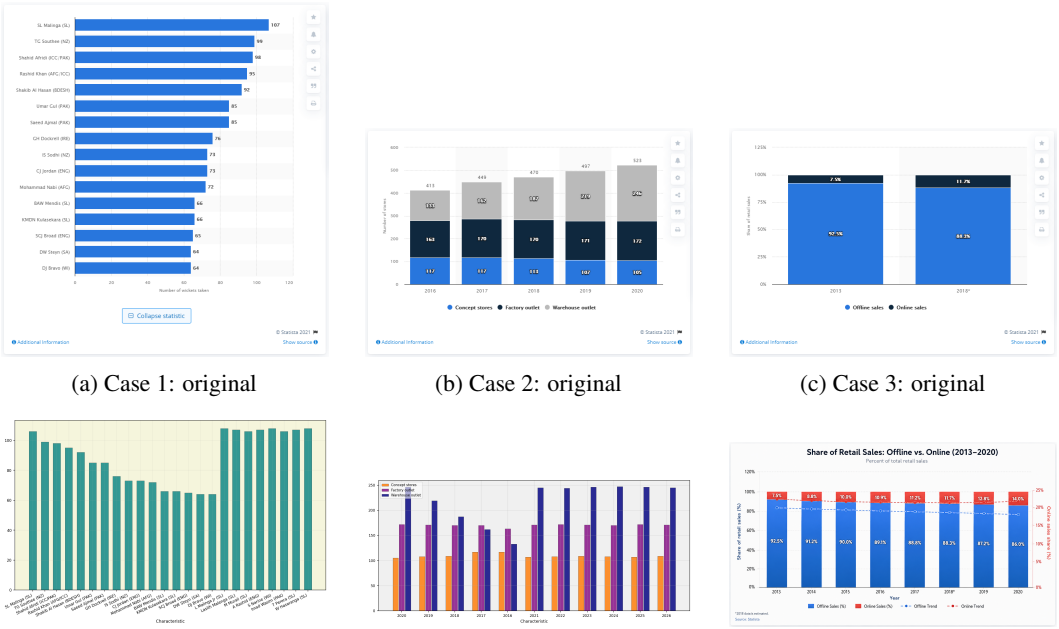
$$a' = \text{table}'[\text{entity} = e^\star, j^\star].$$

This avoids relying on the original row index after rows are inserted or reordered.

For Tier 2 samples, the answer has no unique cell mapping, such as no match or multiple matches. These cases often include derived quantities such as ratios, differences, or percentages. For Tier 2, the EnvRewriter keeps CSV values fixed and only varies rendering style or other answer-preserving visual factors, so the label remains $a' = a$ by construction. In both tiers, no human relabeling or LLM judgment is used.

D.3 ZwZ Answer-Preserving Regional Editing Examples

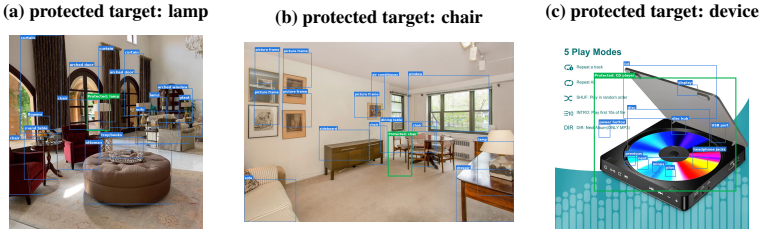
Figure 7 shows representative ZwZ examples. Unlike CLEVR and ChartQA, ZwZ does not provide a complete symbolic state for deterministic relabeling. Thus, VICO keeps the original question-answer



(a) Case 1: original (b) Case 2: original (c) Case 3: original

(d) Case 1: edited (e) Case 2: edited (f) Case 3: edited

Figure 6: **ChartQA edits increase difficulty through table-level changes.** Each column shows one case before and after editing. The EnvRewriter modifies the CSV-level representation and re-renders the chart. For Tier 1 examples, a' is recovered by entity-anchored lookup in the edited table; for Tier 2 examples, CSV values are kept fixed and $a' = a$ is preserved by construction.



green: answer-relevant
blue: other SAM3 regions

green: answer-relevant
blue: other SAM3 regions

green: answer-relevant
blue: other SAM3 regions

Figure 7: **ZwZ edits preserve answers by protecting target objects.** Green boxes mark answer-relevant targets, and blue boxes show other SAM3 regions. VICO keeps $(q' = q, a' = a)$ fixed and edits only non-target regions, allowing background difficulty to change while preserving the original answer.

pair fixed, i.e., $q' = q$ and $a' = a$. We use SAM3 [4] to identify object-level regions and mark the answer-relevant target objects as protected. The EnvRewriter then edits only non-target regions, such as background areas or distractor objects, to adjust visual difficulty while preserving the evidence needed to answer the original question.

These examples highlight the difference between structured editing and natural-image regional editing. For CLEVR, labels are recomputed by executing the original question program on the edited scene graph. For ChartQA, labels are handled by a two-tier rule: Tier 1 computes a' by entity-anchored lookup in the edited CSV table, while Tier 2 preserves $a' = a$ by keeping CSV values fixed and only changing answer-preserving visual factors. For ZwZ, labels are preserved by protecting answer-relevant regions and editing only non-target visual context.

D.3.1 What Happens without Answer Preservation?

A natural alternative is to directly ask a generative image model to produce an easier or harder image. However, this often breaks answer preservation. Figure 8 shows one example. The original question

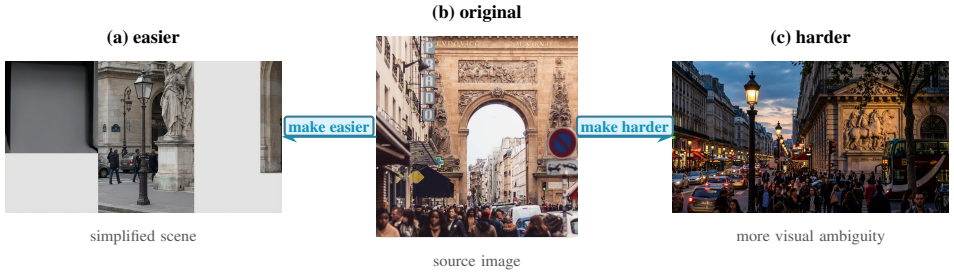


Figure 8: **Direct Nano Banana editing can break answer validity.** Starting from the original image, direct generation can produce easier or harder variants, but the edits may change target relations or introduce ambiguous objects. As a result, the original question-answer pair may no longer be verifiable.

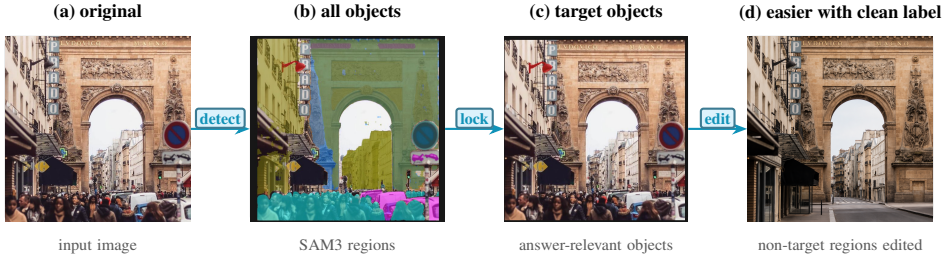


Figure 9: **VICO makes ZwZ easier while preserving the answer.** The pipeline detects object regions, locks the answer-relevant targets, and edits only non-target regions. This changes visual difficulty while keeping the original spatial relation and answer verifiable.

asks: "Where is the street lamp in relation to the sculpture?" The original answer is "left of." After direct generation with Nano Banana, the newly generated answer becomes "right of."

This edit is problematic because the generated image no longer has a clearly verifiable target relation. There are multiple street lamps and multiple sculpture-like objects, so it is unclear which pair the answer refers to. The image may look visually plausible, but the question-answer pair is no longer label-valid. This illustrates why our editing pipeline explicitly enforces answer preservation: the edited image should change visual difficulty while keeping the answer-relevant objects and their relation verifiable.

D.3.2 What Improves with VICO

As shown in Figure 9, this example illustrates how VICO edits a ZwZ image while preserving answer validity. Starting from the original image in (d), VICO first identifies all objects in the scene, as shown in (c). It then selects and locks the answer-relevant target objects mentioned in the question, shown in (b). In this example, the target objects are the street lamp and the sculpture, whose relative relation determines the answer.

After locking these target objects, VICO edits only the non-target regions. This allows the model to simplify the scene by removing distracting content, such as the crowd and traffic signs, while keeping the answer-relevant objects unchanged. The final edited image is shown in (a). The task becomes visually easier, but the answer remains clean and verifiable because the key objects and their spatial relations are preserved throughout the editing process.

E Theoretical Analysis and Saturation

This appendix presents a simplified model that motivates difficulty calibration. It separates the exact variance identity for a binary reward from assumptions about gradient magnitude and a phenomenological cost of distribution drift. The resulting optimum belongs to this model; it is not a proof of optimal task difficulty or convergence for long-form VLM GRPO with format checks, nonstationary data, and imperfect verification.

E.1 Binary Reward Contrast as a Curriculum Motivation

For a binary reward with success probability p, its variance is exactly

$$G(p) = \mathrm{Var}(R) = p(1-p), \tag{E.1}$$

which is maximized at $p = 0.5$. This identity motivates seeking rollout groups containing both successes and failures. Under independent Bernoulli rollouts, the probability that a group of size K contains both outcomes is $1 - p^K - (1-p)^K$, also maximized at 0.5 for $K > 1$. Neither statement alone determines the policy gradient norm or the subsequent test-accuracy gain. For the $\{-1, +1\}$ Actor reward used in training, the corresponding reward variance is $4p(1-p)$, with the same maximizer.

Derivation of Eq. E.1. Let $y \sim \pi_\theta(\cdot \mid x)$ be a sampled solution, and let the verifier return a binary reward $R(y) \in \{0, 1\}$. Define the actor pass rate as

$$p = \Pr(R = 1).$$

Because R only takes values 0 and 1, it follows a Bernoulli distribution:

$$R \sim \mathrm{Bernoulli}(p), \qquad \mathbb{E}[R] = p.$$

Using a REINFORCE estimator with the mean-reward baseline $b = p$, we have

$$\nabla_\theta J(\theta) = \mathbb{E}[(R-p)\,\nabla_\theta \log \pi_\theta(y \mid x)]\,.$$

Since $R \sim \mathrm{Bernoulli}(p)$, we split the expectation into success and failure cases:

$$\begin{aligned}\nabla_\theta J(\theta) = {} & \Pr(R{=}1)(1-p)\,\mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}1] \\ & + \Pr(R{=}0)(0-p)\,\mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}0]\,.\end{aligned}$$

Using $\Pr(R{=}1) = p$ and $\Pr(R{=}0) = 1-p$, we obtain

$$\begin{aligned}\nabla_\theta J(\theta) = {} & p(1-p)\,\mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}1] \\ & + (1-p)(-p)\,\mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}0]\,.\end{aligned}$$

Factoring out $p(1-p)$ gives

$$\nabla_\theta J(\theta) = p(1-p)\,(\mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}1] - \mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}0])\,.$$

Defining

$$\Delta = \mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}1] - \mathbb{E}[\nabla_\theta \log \pi_\theta(y \mid x) \mid R{=}0]\,,$$

we conclude

$$\nabla_\theta J(\theta) = p(1-p)\Delta. \tag{E.2}$$

This factorization does not make Δ independent of p or of the edited task. If $\|\nabla_\theta \log \pi_\theta(y \mid x)\| \le M$, it implies the upper bound $\|\nabla_\theta J\| \le 2Mp(1-p)$, not equality up to a task-independent constant. For the estimator $g = (R-p)\nabla_\theta \log \pi_\theta(y \mid x)$, the same assumption gives $\mathbb{E}\|g\|^2 \le M^2 p(1-p)$; maximizing this bound does not prove that its left-hand side is maximized at $p = 0.5$. Moreover, $b = p$ is a valid baseline but is not generally the variance-minimizing baseline when score norms depend on the outcome. We therefore use $p(1-p)$ as a reward-contrast proxy rather than an exact characterization of GRPO learning progress.

Finite-rollout estimation. With $K = 8$, $\hat{p}$ has increments of $1/8$, so the band $[0.35, 0.65]$ retains observed pass rates 0.375, 0.5, and 0.625. Under independent sampling, $\mathrm{Var}(\hat{p}) = p(1-p)/8$; an individual edit can therefore be misclassified relative to the target band. Temperature 0.7 and top-$p = 1.0$ permit stochastic responses, but do not guarantee useful diversity for every task. The target is a practical design choice, not an exact measurement of task difficulty. Related motivations appear in GenEnv's α-curriculum [14] and VisPlay's uncertainty reward [16]; these similarities do not establish a universal optimum across reward designs. V-Zero [36] further illustrates this distinction: it uses $\min(c, 1-c)$ when the reasoned answer agrees with an intuitive answer, and $0.5c$ otherwise. Vote confidence c and verified Actor success probability p are different quantities, so their difficulty objectives should not be equated.

E.2 Distribution Drift Cost

The variance proxy does not account for the cost of changing a task. Visual edits that substantially alter difficulty may also alter the training distribution, creating features that do not transfer to the test set. This is a possible cost in both visual and textual generation; we do not assume it is absent in text-based systems. We model the tradeoff with a quadratic penalty on the change in pass rate. This penalty is a phenomenological proxy rather than a measured distance between image distributions.

We define a proxy utility for a round as a reward-contrast term minus a distribution-drift penalty:

$$U_t(\hat{p}) = C_t \cdot G(\hat{p}) - \lambda \cdot D_t(\hat{p}), \tag{E.3}$$

where $C_t > 0$ models remaining learning headroom, D_t models a drift cost, and $\lambda > 0$ sets its weight. These quantities are not estimated from the reported runs, and U_t is not identified with measured accuracy gain. The unpenalized proxy corresponds to $D_t \equiv 0$.

We adopt the simplest convex form for the drift cost:

$$D_t(\hat{p}) = \kappa_t \cdot (\hat{p} - \hat{p}_{\text{orig},t})^2, \tag{E.4}$$

which is zero when the pass rate is unchanged and grows with its deviation from the original rate. Different edits can have equal pass rates but different semantic or distributional effects, so this proxy cannot measure those effects individually.

E.3 Optimal Difficulty Target

Substituting Eqs. E.1 and E.4 into Eq. E.3 and taking the derivative with respect to $\hat{p}$,

$$\frac{dU_t}{d\hat{p}} = C_t(1 - 2\hat{p}) - 2\lambda\kappa_t(\hat{p} - \hat{p}_{\text{orig},t}) = 0.$$

Solving yields

$$\hat{p}_t^* = \frac{C_t + 2\lambda\kappa_t \cdot \hat{p}_{\text{orig},t}}{2C_t + 2\lambda\kappa_t}. \tag{E.5}$$

Defining $\rho_t = \frac{\lambda\kappa_t}{C_t + \lambda\kappa_t} \in [0, 1]$, this rearranges to a convex combination:

$$\boxed{\hat{p}_t^* = \tfrac{1}{2} + \rho_t \cdot \left(\hat{p}_{\text{orig},t} - \tfrac{1}{2}\right).} \tag{E.6}$$

The optimal target is a weighted interpolation between 0.5 and the current $\hat{p}_{\text{orig},t}$, with the weight ρ_t determined by the ratio of drift cost to remaining capacity.

E.4 Two Limiting Regimes

Equation E.6 unifies two extremes.

Early-stage ($\rho_t \to 0$, i.e., $C_t \gg \lambda\kappa_t$). The Actor has ample capacity headroom, and light editing suffices to reach the target zone. The optimal target approaches 0.5, recovering the gradient-signal optimum. In this regime, difficulty calibration should be aggressive.

Saturation regime ($\rho_t \to 1$, i.e., $C_t \to 0$). Within the proxy model, shrinking C_t makes the drift term relatively more influential. The utility-maximizing target approaches $\hat{p}_{\text{orig},t}$, suggesting less aggressive difficulty adjustment. This limiting case describes the model's tradeoff; it does not establish that a measured Actor has reached a capacity ceiling.

E.5 From Theory to the Sweet-Zone Reward

In practice, we do not estimate C_t or κ_t. We use the fixed band $[0.35, 0.65]$ and reward $R_R(\hat{p}) = \exp\left(-\beta(\hat{p} - 0.5)^2\right)$ to favor intermediate observed difficulty. This objective does not implement the adaptive optimum in Eq. E.6, nor guarantee that the optimum remains inside the band. The experiments below describe the behavior of the fixed-target implementation.

Table 14: **Sweet-zone coverage declines across rounds.** The fraction of EnvRewriter trajectories in $[0.35, 0.65]$ falls from 53.4% to 11.1%: a decrease of 42.3 percentage points, or 79.2% relative to R1. This measures alignment with the fixed difficulty band, not the optimal target in Eq. E.6.

Round	Sweet-zone hit rate (%)
R1	53.4
R2	29.5
R3	21.2
R4	20.5
R5	11.1

E.6 Empirical Saturation Evidence

The reference ChartQA run improves from 76.90% at the base model to 79.84%, 82.24%, 86.96%, and 89.12% at R1–R4, respectively, before decreasing slightly to 88.94% at R5. The round-over-round gains vary rather than decrease monotonically. The final 0.18-percentage-point decline does not by itself establish a capacity ceiling or identify distribution drift as its cause. Multi-seed results for R1–R4 are reported in Appendix H.5.

Table 15: **ChartQA accuracy across co-evolution rounds.** Accuracy improves through R4, with varying round-over-round gains, and decreases slightly at R5. Δ denotes the change from the previous round in percentage points.

Round	ChartQA (Qwen3-VL-8B)	Δ
Base	76.90	—
R1	79.84	2.94
R2	82.24	2.40
R3	86.96	4.72
R4	89.12	2.16
R5	88.94	-0.18

Observed pass rates move away from the target. The mean edited pass rate decreases from 0.502 at R3 to 0.375 at R5. This measures the difficulty of actual edits, not the optimal target in Eq. E.6. At R5 the original pass rate is 0.889, so the model's optimum lies between 0.5 and 0.889; the observed decrease below 0.5 does not validate movement toward that optimum. It instead indicates that the generated samples and the fixed target become less well aligned.

Fewer edits meet the EnvRewriter's difficulty filter. The fraction of edits in $[0.35, 0.65]$ decreases from 53.4% at R1 to 11.1% at R5. This reduces the fraction eligible for EnvRewriter updates. It does not imply that all out-of-band samples are excluded from Actor training or that their gradients have been measured to vanish.

Per-round gains vary, with a slight decline at R5. The reference gains are $+2.94$, $+2.40$, $+4.72$, $+2.16$, and -0.18 percentage points for R1–R5, respectively. The largest round-over-round improvement occurs at R3. R4 provides a further improvement, bringing the cumulative gain over the base model to 12.22 percentage points, while R5 yields no additional improvement in this reference run. These observations characterize the behavior of the fixed-target curriculum but do not establish a general saturation pattern or its cause. They do not provide estimates of C_t, κ_t, or the Actor's remaining capacity.

E.7 Pass-Rate Distribution Shift

The reported sweet-zone frequencies show that a smaller fraction of edits falls in the target band at later rounds. Means and band frequencies should be read together: an average near 0.5 can conceal a mixture of near-zero and near-one pass rates. The available summaries do not identify the full distribution or establish a causal explanation for test-set saturation.

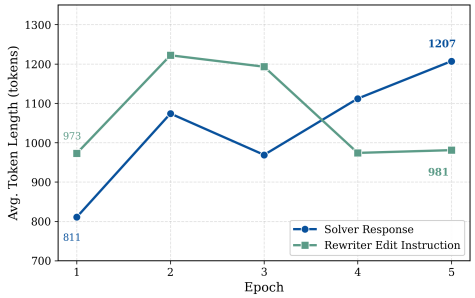


Figure 10: **EnvRewriter edit length stays stable.** Multi-round training improves edit strength without making edit instructions longer.

F Evaluation Protocols and Prompt Templates

F.1 Evaluation Benchmarks and Metrics

We use a common zero-shot protocol for our trained models and in-house ablations. Published reference scores retain the provenance described in Appendix F.6; they are not all independent reruns in our harness. For mathematical reasoning, we use MathVista, MathVision, MathVerse, WeMath, LogicVista, and ChartQA. For vision-centric understanding, we use MMMU, MMVP, and BLINK. Unless otherwise noted, the average in the main tables is the equally weighted average of the mathematical-reasoning average and the vision-centric average.

F.2 Answer Normalization and Scoring

For rule-based verification, the model response is first parsed into the required answer field. We normalize trivial formatting differences, including surrounding whitespace, capitalization for short textual answers, and common numerical formats. For ChartQA, we follow the official relaxed numerical accuracy protocol when reporting ChartQA-specific accuracy. For other benchmarks, we use each benchmark's standard answer matching or official evaluation script whenever available.

F.3 Actor Rollout Prompt

The Actor is prompted to solve the visual question and return a response in the required reasoning-and-answer format. The key requirement is that the final answer can be parsed reliably by the verifier.

> **Actor prompt template.** Given the image and the question, reason step by step and provide the final answer in the required format. Put the final answer inside `<answer>` and `</answer>`. The answer should be concise and directly match the question.

F.4 EnvRewriter Prompt

The EnvRewriter is prompted to modify the visual environment while preserving verifiability. For structured domains, it must output an executable edit command over the scene graph or data table. For natural-image VQA, it must avoid changing the protected answer-relevant region.

> **EnvRewriter prompt template.** You are an environment rewriter. Given the original image metadata, question, answer, current Actor pass rate, and available edit tools, propose one valid edit that adjusts task difficulty toward the learnable zone. Return a structured command with the fields `tool`, `params`, and `reasoning`. The edit must remain executable and must preserve or programmatically update the answer.

F.5 Evaluation Prompt

During evaluation, we use a deterministic prompt format and do not expose the model to EnvRewriter metadata, pass-rate statistics, or training-time edit commands. The model only receives the test image and question.

Table 16: **Base-model accuracy depends on the evaluation protocol.** All rows concern Qwen3-VL-8B-Instruct. The main reference is sourced from MM-Zero; the other rows are three additional VLMEvalKit runs. They are reported separately and must not be combined to estimate a training gain across protocols. The final row averages the three displayed runs.

Protocol	ChartQA	MathVerse	MathVista	LogicVista
Main reference (MM-Zero)	76.9	50.6	67.7	55.1
VLMEvalKit, run 1	82.8	38.2	74.8	57.7
VLMEvalKit, run 2	83.3	38.1	75.2	57.0
VLMEvalKit, run 3	83.0	38.7	75.1	57.3
Mean	83.0	38.3	75.0	57.3

Evaluation prompt template. Answer the question based on the image. Provide the final answer in `<answer>` and `</answer>`. Do not include any information that is not supported by the image.

F.6 Baseline Implementation Details

Published references and in-house ablations. The Qwen3-VL-8B-Instruct reference scores in the main comparison are sourced from MM-Zero [22], which provides reported scores and public correct/total logs for overlapping benchmarks. Other published baselines use their reported results where available. These comparisons provide context, but differences in prompting, parsing, answer extraction, decoding, and judging can affect cross-paper rankings. In particular, a common model checkpoint does not imply a common evaluation protocol.

For the in-house Direct GRPO, Supervised GRPO, text-only co-evolution, static offline augmentation, and VICO comparisons, we use the same Actor backbone, seed data source, verifier, decoding configuration, and evaluation protocol within each domain. Rollout groups contain eight responses; temperature is 0.7, top-p is 1.0, and the maximum response length is $3{,}824$ tokens. Matching these settings controls important implementation differences, but does not establish equal total compute or equal training exposure. Multi-round generation and EnvRewriter optimization add cost; the available resource accounting is given in Appendix G.3.

F.7 Evaluation Protocol Sensitivity and Generation Budget

Formatting and extraction requirements can affect a base Instruct model and an RL-trained model differently. A uniform harness is therefore useful for comparison, but is not by itself evidence that every gain reflects reasoning rather than improved format compliance. To examine sensitivity, we separately evaluate the base Qwen3-VL-8B-Instruct checkpoint with VLMEvalKit. Table 16 reports three such runs alongside the main reference values. The differences across benchmarks show why scores obtained under different protocols must be reported separately. This base-only diagnostic does not establish the size of VICO's gain under the alternate protocol.

The $3{,}824$-token response limit applies to both rollout sampling and evaluation in our common in-house setup. Equal limits avoid allocating a larger generation budget to VICO, but can still constrain methods with different response-length distributions.

Longer-generation diagnostic. In an additional VLMEvalKit sensitivity experiment, increasing the maximum generation length from $8{,}192$ to $32{,}768$ tokens improves `MathVerse_MINI_Vision_Only` by 0.38 percentage points and `MathVision_MINI` by 0.33 points. The small changes indicate limited benefit from extending this particular budget on these two evaluated splits. They do not measure the effect of increasing the main $3{,}824$-token limit, or establish that all models and benchmarks are insensitive to truncation. Reproduction requires the exact checkpoint, prompt, parser, split, decoding configuration, and treatment of malformed or truncated responses.

F.8 Full ChartQA EnvRewriter Prompt

```
You are an expert chart editing planner.
```

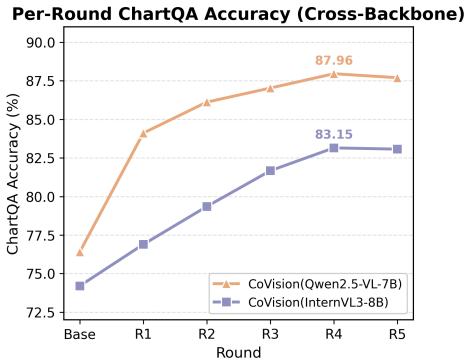


Figure 11: **VICO improves accuracy across VLM backbones.** Per-round ChartQA accuracy rises for both tested models, showing that the emergent curriculum is not tied to one model family.

```
Goal: choose one edit that moves the Actor pass rate (p̂) closer to 0.5.

Tool:
- chart_rerender(edit_instruction): edit the chart's underlying data table and
  re-render it with matplotlib.

Rules:
- To make harder: add near-miss distractors, change the answer value, or increase
  chart density.
- To make easier: remove distractors, simplify the chart, or make the answer
  more distinct.
- Return one specific executable edit.

Output format:
<think>
your reasoning
</think>
<tool_call>
{"tool": "chart_rerender", "params": {"edit_instruction": "your edit"}}
</tool_call>
```

Listing 1: rewriter editing planner prompt.

G Additional Main Results

G.1 Full OOD Average Calculation

For the Chart-domain variant, we report the out-of-distribution average by excluding ChartQA, because ChartQA is the in-domain visual source for that variant. This avoids overstating generalization by including the benchmark most directly aligned with the training domain. For CLEVR and ZwZ variants, the same benchmark list is kept for consistency in the main comparison.

G.2 Cross-Backbone Results

VICO improves across different VLM backbones, suggesting that the co-evolution loop is not specific to a single model family.

G.3 Cost and Annotation Efficiency

We separate recurring QA annotation, editable metadata and tooling, and training compute. These resources have different units and should not be collapsed into a single claim of data or compute efficiency.

Definition of label cost. We define labeled QA pairs as human-annotated or teacher-generated question-answer supervision. The chart setting begins with 2,000 existing seed examples and their tables. During co-evolution, new labels are computed from edited structures or retained under the preservation rules, without additional human- or teacher-written QA labels. This does not make the seed annotations free, and it does not count the GPT-5 editing demonstrations used to initialize the

Table 17: **Compute breakdown for the main 8B Chart setting.** Approximate A100 GPU-hours for initialization and co-evolution through R4. The sum includes EnvRewriter training; R5 is a separate saturation experiment. CPU rendering is listed separately from GPU time.

Component	**A100 GPU-hours**
EnvRewriter initialization SFT	5
EnvRewriter RWR updates	21.25
Actor rollout generation	40
Actor GRPO, R1–R4	305
Total	**371.25**
Chart rendering / editing	CPU; negligible GPU

Table 18: **Reported compute for the Chart training pipelines.** Costs are approximate A100 GPU-hours. The 8B VICO total includes EnvRewriter initialization and updates, rollouts, and Actor GRPO through R4. These are measured pipeline budgets, not equal-compute comparisons.

Method	**Actor backbone**	**A100 GPU-hours**
Direct GRPO	Qwen2.5-VL-7B	~ 55
VICO	Qwen2.5-VL-7B	~ 149
Direct GRPO	Qwen3-VL-8B	~ 60
VICO (R1–R4)	Qwen3-VL-8B	~ 371.25

EnvRewriter as zero-cost supervision. Those demonstrations teach edit commands and strategies, rather than provide new QA targets, and must be accounted for separately.

Metadata and tooling. The provided-structure experiments use dataset CSV tables or scene graphs, and the natural-image arm requires reliable target grounding. Blender, matplotlib, segmentation, and image-editing backends are additional dependencies. Recovering a structure with ChartMoE reduces reliance on provided tables but introduces a learned extraction tool with its own pretraining and inference costs. Thus, the $16\times$–$160\times$ comparison with chart-specialized methods describes seed-count versus labeled-QA-count ratios, not a reduction by the same factor in total supervision, metadata, generated training examples, or compute.

Compute budget and accounting boundary. The main Qwen3-VL-8B Chart run requires approximately 371.25 A100 GPU-hours through R4: 5 for EnvRewriter initialization SFT, 21.25 for EnvRewriter RWR, 40 for Actor rollout generation, and 305 for Actor GRPO (Table 17). Thus, the 345-hour Actor-plus- rollout subtotal excludes 26.25 hours of EnvRewriter training. Chart rendering is CPU-based with negligible GPU use; CPU time and external API charges are not converted to A100 GPU-hours. R5 is an additional saturation experiment outside this R1–R4 accounting boundary. The stage totals do not separately itemize failed generations or retries.

Baseline budgets and shared settings. Direct GRPO requires approximately 60 A100 GPU-hours in the 8B Chart setting. For Qwen2.5-VL-7B Chart, the reported totals are approximately 149 for VICO and 55 for Direct GRPO. Table 18 summarizes these paired compute totals. Common backbones, seed sources, rollout settings, verifiers, and evaluation protocols control implementation differences; they do not make total training exposure or compute equal. Exact baseline update counts and the unreported stage totals are not inferred from these aggregate costs. Separate totals for Supervised GRPO, text-only co-evolution, and static augmentation are not included in this accounting.

Interpreting the tradeoff. VICO spends additional generation and optimization effort to adapt the visual curriculum while avoiding recurring QA annotation. The comparison with more data-intensive methods does not establish compute superiority: reported GPU-hours may use different hardware, training stages, or accounting boundaries. The monetary cost of an alternative annotation pipeline also depends on its verification procedure and required label accuracy. We therefore make the narrower claim of human-QA-label efficiency under pre-existing or recoverable verifiable structure.

No additional human annotation during training. The zero-additional-label claim applies to QA supervision during the co-evolution loop. It does not imply zero human effort throughout the

Table 19: **VICO cuts labeling cost while staying competitive.** Label Cost counts teacher or judging LLM tokens with the Qwen2.5 tokenizer. Programmatic-label methods, including VICO, use no LLM labeling tokens; baseline training costs are estimated from reported RL samples, while VICO cost is directly measured.

Method	Data Cost			Training			Performance	
	Prepare Method	Num (RL)	Label Cost (Tokens)	Method	Interact	Time Cost	MathVerse	Mathvista
Qwen2.5-VL-7B	–	–	–	–	–	–	39.6	68.2
R1-OneVision-7B	Programmatic + human checks	10k	$\geq$ 1.1 M	SFT+GRPO	✗	$\geq$ 170 A100-Hours	46.4	64.1
VLAA-Thinker-7B	Teacher LLM-generated traces	25k	29.6 M	SFT+GRPO	✗	$\geq$ 120 A100-Hours	48.2	68.0
OpenVLThinker-7B	Teacher LLM-generated solutions	9k	5.7 M	SFT+GRPO	✗	$\geq$ 125 A100-Hours	47.9	70.2
MM-Eureka-Qwen-7B	Human-curated	15k	–	GRPO	✗	$\approx$ 700 A100-Hours	50.3	73.0
ViGaL-Snake	Game environment via PPO	72k	0	RLOO	✗	$\approx$ 170 A100-Hours	51.1	70.7
ViGaL-Rotation	Game environment via PPO	72k	0	RLOO	✗	$\approx$ 170 A100-Hours	50.4	71.2
VICO-Qwen2.5VL-7B(Chart)	**Co-evolutionary editing**	**2000**	**0**	**Co-evolve+GRPO**	✓	**$\approx$ 149 A100-Hours**	**54.9**	**72.8**

Table 20: **VICO reaches competitive ChartQA accuracy with far fewer labeled QA pairs.** All methods use ChartQA's official relaxed accuracy with 5% numerical tolerance. VICO uses seed inputs and avoids extra labeled QA supervision, while chart-specialized baselines require tens or hundreds of thousands of labeled pairs.

Method	Labeled QA	ChartQA (%)
Chart-R1-7B [6]	$\sim$33k	91.04
Bespoke-MiniChart-7B [30]	$\sim$91k	89.5
ECD (Qwen2.5-VL-7B) [50]	$\sim$320k	85.32
VICO-Qwen2.5-VL-7B (Ours)	2,000 seed inputs	87.96
VICO-Qwen3-VL-8B (Ours)	2,000 seed inputs	89.12

project: the independent manual validity audit, seed construction, and tool development serve separate purposes.

G.4 Comparison with Chart-Specialized Methods

Table 20 compares VICO against three recent chart-specialized methods: Chart-R1 [6], Bespoke-MiniChart-7B [30], and ECD [50]. VICO-Qwen3VL-8B reaches 89.12% on ChartQA using seed visual inputs and no additional labeled QA pairs, while remaining competitive with methods that use tens or hundreds of thousands of labeled pairs.

H Ablation and Robustness Studies

H.1 EnvRewriter SFT as a Co-evolution Catalyst

We examine the relationship between supervised initialization and co-evolution using four experimental settings (Table 6). The EnvRewriter is initialized on verified GPT-5 edit demonstrations, which supervise both executable command generation and difficulty-adjustment strategies. These comparisons assess the benefit of combining this initialization with co-evolution.

Direct Actor SFT baseline. Applying supervised fine-tuning directly to the Actor, without co-evolution, improves ChartQA from 76.9% to 79.4%, a gain of 2.5 percentage points. This comparison measures the effect of direct Actor SFT in the evaluated setting; it does not isolate the contribution of the GPT-5 demonstrations used to initialize the EnvRewriter.

Co-evolution without EnvRewriter SFT. Running the co-evolution loop with a vanilla, non-SFT EnvRewriter reaches 78.6% (1.7 percentage points above the base model). On manual inspection, the vanilla EnvRewriter frequently produces malformed or unexecutable edit commands, limiting the supply of valid training samples. This result highlights the importance of initializing an EnvRewriter that can produce executable edits.

Combining EnvRewriter SFT and co-evolution. Initializing the EnvRewriter with verified GPT-5 edit demonstrations and then updating it through co-evolution yields 89.1% on ChartQA (12.2 percentage points above the base model). The combined setting outperforms both direct Actor SFT and co-evolution with a non-SFT EnvRewriter, supporting the complementary roles of supervised initialization and iterative actor–environment adaptation.

These results support combining EnvRewriter SFT with co-evolution. Because the initialization demonstrations supervise both command format and editing strategy, the present comparisons do not separately quantify these two contributions or isolate the contribution of GPT-5-generated supervision.

H.2 Static Offline Augmentation Details

To test whether the gain comes simply from stronger generated data, we compare VICO against static offline augmentation without the iterative actor–environment loop. The offline variants generate a fixed corpus with self-generation, GPT-5, or Nano Banana, and then train the Actor on that fixed data. These variants improve over the base model but remain below iterative visual co-evolution, showing that the dynamic feedback loop contributes more than one-shot data quality.

H.3 Text-Only Editing vs. Visual Editing

To isolate the role of the editing modality, we compare text-only perturbation with visual editing under the same CLEVR training source. Text-only perturbation changes the question or answer format while keeping the visual scene fixed. Visual editing modifies the image itself through the verifiable editing pipeline. The main results show that visual editing provides larger gains, especially on chart reasoning, supporting the claim that effective multimodal co-evolution requires adapting the visual environment rather than only rewriting text.

H.4 Robustness to Imperfect Image Editing

In practice, image editors may produce failures such as blank renders or mismatches between images and labels. To simulate partial editing failures, we corrupt 20% of the R1 training data: 10% of the samples are replaced with blank white images, and another 10% use unedited original images paired with edited labels. As shown in Table 7, the overall average performance drops from 66.8% to 66.0%, a decrease of 0.8 percentage points, and the noisy model still outperforms the base model by 2.4 percentage points. This shows that VICO does not rely on a perfect image editor: failed edits do not fully destroy the training signal, and the remaining valid samples still support stable co-evolution.

H.5 Stability Across Random Seeds

A natural concern for any RL-based training pipeline is whether the reported gains depend on a favorable random seed. We re-ran the first round of training with three independent random seeds on CLEVR and evaluated the resulting VICO-8B (CLEVR) checkpoints on ChartQA and MathVista.

Table 21: **VICO is stable across random seeds.** We re-run the first CLEVR training round with three seeds and evaluate on ChartQA and MathVista. Standard deviation is far below the reported gains, so improvements are not driven by favorable seed selection.

Seed	ChartQA (%)	MathVista (%)
Seed 1 (reported)	85.2	73.8
Seed 2	84.5	72.9
Seed 3	85.2	71.7
Mean $\pm$ Std	85.0 ± 0.4	72.8 ± 1.1
Base (Qwen3VL-8B)	76.9	67.7
Gain over base	$+8.1$	$+5.1$

The CLEVR experiment measures first-round variability; it does not establish that later rounds are independent of initialization. To examine propagation across the complete loop, we additionally run the ChartQA setting through R4 with three independent seeds (Table 22). All three runs improve at every round from R1 to R4. The R4 mean is 88.89% with a sample standard deviation of 0.20 percentage points, approximately 12.0 points above the 76.90% base reference. These results support stability for the tested chart setting; they do not establish multi-round stability for every domain or backbone. R5 is reported for the reference trajectory and is not covered by this three-seed comparison.

Table 22: **ChartQA improvements persist through R4 across three seeds.** Accuracy is reported in percent for three complete Chart-domain runs. Mean and sample standard deviation are computed across the three seeds at each round; the base-model reference is 76.90%.

Round	Seed 1	Seed 2	Seed 3	Mean	Std.
R1	85.24	84.48	85.04	84.92	0.39
R2	87.50	87.20	87.16	87.29	0.19
R3	88.47	87.72	87.88	88.02	0.40
R4	89.12	88.76	88.80	88.89	0.20

Table 23: **The EnvRewriter shifts Actor pass rates in both directions.** The signed shift is $\hat{p}_{\text{orig}} - \hat{p}_{\text{final}}$: positive values indicate a lower pass rate after editing, and negative values indicate a higher pass rate. This signed quantity differs from the nonnegative editing intensity $|\hat{p}_{\text{orig}} - \hat{p}_{\text{final}}|$. Rounding can affect the difference between the displayed pass rates and shifts.

Round	n	$\hat{p}_{\text{orig}}$	$\hat{p}_{\text{final}}$	Signed shift
ChartQA (strong Actor → EnvRewriter pushes down)				
R1	2200	0.846	0.622	+0.225
R2	2200	0.829	0.502	+0.328
R3	2200	0.822	0.502	+0.321
R4	2200	0.848	0.387	+0.461
R5	2200	0.889	0.375	+0.514
ZwZ (weak Actor → EnvRewriter pulls up)				
R1	2000	0.209	0.651	-0.442
R2	2000	0.229	0.583	-0.354
R3	2000	0.223	0.524	-0.301
R4	2000	0.231	0.433	-0.203
R5	2000	0.227	0.442	-0.215

I Co-Evolution Dynamics and Qualitative Analysis

I.1 Bidirectional Difficulty Calibration

The EnvRewriter calibrates difficulty from both directions. On ChartQA, the initial Actor is strong, so the EnvRewriter lowers the pass rate toward the sweet zone. On ZwZ, the initial Actor is weak, so the EnvRewriter raises the pass rate toward the sweet zone. This supports the claim that VICO performs actor-relative calibration rather than random perturbation.

What the calibration statistics establish. Because both policies change across rounds, success rates measured with the current Actor are actor-relative. A larger pre/post-edit pass-rate shift does not by itself establish that later images are intrinsically harder for a fixed model. Likewise, a mean pass rate near 0.5 can arise from a mixture of trivial and unsolved samples. We therefore report sweet-zone hit rates in addition to mean pass rates.

I.2 Difficulty under a Frozen Base Model

We evaluate edited trajectory pools from generation rounds R0–R4 using the same frozen Qwen3-VL-8B base model throughout (Table 24). Mean pass rate decreases from 0.401 at R0 to 0.295 and 0.299 at R2 and R3, respectively, while mean difficulty, defined as $1 - \text{mean}(\hat{p})$, increases from 0.599 to approximately 0.70. Because the evaluator is fixed, these differences cannot be attributed solely to changes in the evaluated Actor checkpoint. They support a difficulty shift that transfers to a fixed base evaluator, beyond the actor-relative statistics above.

The trend is not monotonic: mean pass rate rebounds to 0.360 at R4, which remains below the R0 value. The rebound does not by itself identify the cause of saturation or rule out every form of actor-specific overfitting. These are generation-round diagnostics, distinct from the held-out test-accuracy trajectories and three-seed training results.

Table 24: **Edited-sample difficulty under a fixed evaluator.** The Qwen3-VL-8B base model is held fixed while evaluating edited samples from successive generation rounds. The mean and median summarize sample pass rates, and mean difficulty is $1 - \text{mean}(\hat{p})$. Round labels identify the trajectory pools, not the test-accuracy checkpoints or independent training seeds.

Generation round	Mean $\hat{p}$	Median $\hat{p}$	Mean difficulty
R0	0.401	0.375	0.599
R1	0.376	0.250	0.624
R2	0.295	0.125	0.705
R3	0.299	0.125	0.701
R4	0.360	0.250	0.640

I.3 Editing Intensity and Response Length

Editing intensity and output length evolve differently. As reported in Table 25 and Figure 12(a), editing intensity increases from 0.225 at R1 to 0.514 at R5, a relative increase of 128.4%, while the average EnvRewriter instruction length remains almost constant (973 → 981 tokens; +0.8%). The Actor's response length grows from 811 to 1207 tokens (+49%). These observations show larger pass-rate shifts and longer Actor responses, but do not by themselves establish better difficulty calibration or greater intrinsic task difficulty: the Actor changes across rounds, and the sweet-zone hit rate decreases.

Table 25: **Editing intensity grows while EnvRewriter instruction length stays nearly fixed.** Between R1 and R5, the reported pass-rate shift and Actor response length increase. Changes are relative to R1; these summaries do not independently measure intrinsic task difficulty.

Metric	R1	R5	Change
Editing intensity $\lvert\hat{p}_{\text{orig}} - \hat{p}\rvert$	0.225	0.514	+128.4%
EnvRewriter instruction length	973	981	+0.8%
Actor response length	811	1207	+49%

Figure 12(b,c) shows that the per-round accuracy gains are +2.94, +2.40, +4.72, +2.16, and −0.18 percentage points at R1–R5, respectively, matching Table 15. The gains do not decrease monotonically: R3 provides the largest improvement, and only R5 shows a slight decline. The sweet-zone hit rate falls from 53.4% to 11.1%, a decrease of 42.3 percentage points (79.2% relative to R1). These diagnostics describe reduced alignment with the fixed difficulty band and a final-round accuracy decline; they do not identify its cause. Their interpretation and limits are discussed in Appendix E.6.

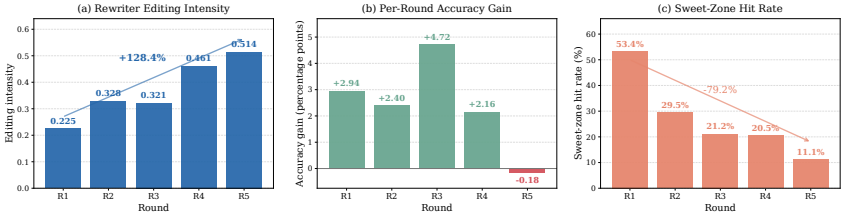


Figure 12: **Editing intensity, accuracy gains, and sweet-zone coverage across rounds.** (a) Editing intensity increases from 0.225 to 0.514 (+128.4% relative to R1). (b) Per-round accuracy gains are +2.94, +2.40, +4.72, +2.16, and −0.18 percentage points, with the largest gain at R3. (c) Sweet-zone hit rate falls from 53.4% to 11.1% (−79.2% relative to R1). These diagnostics do not directly measure the optimal target or remaining model capacity.

I.4 Sample-Level Flip Analysis

Table 26 reports sample-level correctness counts for the base model and R4 Actor on 2,500 ChartQA test samples. The 365 wrong-to-correct flips outnumber the 64 correct-to-wrong regressions by 5.7×, yielding a net gain of 301 correct answers, or 12.04 percentage points. These counts imply a base accuracy of 77.08% and an R4 accuracy of 89.12%. The count-derived base accuracy differs from the 76.90% reference in Table 15; consequently, the 12.04-point count-derived gain and the 12.22-point

Table 26: **Positive capability flips dominate regressions.** On the reported $2{,}500$-sample ChartQA prediction set, 365 wrong-to-correct flips outnumber 64 correct-to-wrong regressions by $\mathbf{5.7}\times$. The counts imply 77.08% Base accuracy, 89.12% R4 accuracy, and a 12.04-percentage-point net gain; the reference trajectory in Table 15 instead uses a 76.90% Base accuracy.

Outcome	**n**	**%**
Both correct (Base ✓, R4 ✓)	1863	74.5
Improvement (Base ✗ → R4 ✓)	365	14.6
Regression (Base ✓ → R4 ✗)	64	2.6
Both wrong (Base ✗, R4 ✗)	208	8.3
Total	**2500**	**100.0**

Table 27: **Reported sample-level ChartQA correctness trajectories.** Each sample is represented by a 5-bit correctness sequence across Base and R1–R4. The first block shows first-correct rounds; the second lists common trajectories. The top five patterns cover 92.5% of samples, led by stable correctness (`11111`) and stable R1 unlocks (`01111`).

State	**n**	**%**
Already correct at Base	1927	77.1
Newly unlocked at R1	347	13.9
Newly unlocked at R2	37	1.5
Newly unlocked at R3	15	0.6
Newly unlocked at R4	3	0.1
Persistent failures	171	6.8
Mutually exclusive sum	**2500**	**100.0**
Non-monotonic (any regression)	197	7.9

Pattern	**n**	**%**
`11111`	1796	71.8
`01111`	306	12.2
`00000`	171	6.8
`10111`	22	0.9
`10000`	18	0.7
Top 5	2313	92.5
Other patterns	187	7.5
Total	**2500**	**100.0**

reference gain are not interchangeable. Table 27 provides the reported correctness trajectories across Base and R1–R4.

I.5 Question-Type Breakdown

The main text shows that gains concentrate on reasoning-heavy ChartQA question types, such as ratio/percentage, value lookup, and difference. These categories require multi-step reading and computation, so their larger improvements support the claim that VICO improves visual reasoning rather than simply increasing recognition accuracy on easy cases.

Scope of MathVista transfer. The results in Table 2 establish benchmark-level transfer on MathVista and the other evaluated mathematical-reasoning tasks. These aggregate results do not establish uniform gains across MathVista categories or independence from similarity to synthetic scenes.

I.6 Chain-of-Thought Case Studies

CoT case analysis. Representative cases show that the base model tends to produce direct, single-step answers that miss key constraints, whereas the trained model exhibits explicit step-by-step decomposition, such as identifying relevant chart regions, reading the y-axis, and computing the final answer. Manual inspection of 20 randomly sampled base-wrong $\rightarrow$ R4-correct pairs confirms this pattern is systematic.

J Relation to Perception Shaping and Static Visual Augmentation

Perception-oriented optimization and visual-environment adaptation act on different parts of the training pipeline. PAPO [40] adds a perception-aware objective to policy optimization. VPPO [18] uses visual dependence to reweight trajectory advantages and focus updates on perceptually relevant

tokens. BiPS [56] constructs question-conditioned evidence-preserving and evidence-ablated views to shape visual reliance. Region-to-image distillation [41] transfers information from local regions to full-image perception. These approaches can modify visual evidence or its use during training; their distinction from VICO is not simply whether an image changes.

VICO learns an edit policy whose training signal depends on the current Actor's measured success rate, and constrains the resulting tasks through domain-specific validity checks. Static offline augmentation instead fixes its generated corpus before Actor training. The ChartQA comparison in Table 5 shows that self-generation, GPT-5, and Nano Banana reach 78.6%, 80.1%, and 81.9%, respectively, compared with 89.1% for VICO in the evaluated settings. This supports the usefulness of adaptation in those experiments; it does not establish superiority over every static augmentation strategy under equal total compute.

The broader self-evolution literature also includes visual generation: MM-Zero [22] jointly trains roles that propose tasks, generate rendering code, and solve the resulting images. Our more specific distinction is constrained editing of existing instances with recomputation or preservation of labels. Likewise, GenEnv [14] already studies difficulty-aligned agent–environment co-evolution; VICO develops this idea for visual editing interfaces with explicit supervision constraints.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This work uses public visual reasoning benchmarks and does not involve human subjects, private personal data, or deceptive data collection. The research is intended for benchmark-style VLM reasoning and follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations related to verifiable image-side structures, imperfect image editing, compute overhead, and saturation in later co-evolution rounds. These limitations are addressed in the Limitations and Impact Statement and further supported by the robustness, compute, and saturation analyses in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper includes a theoretical analysis of difficulty calibration, including the gradient-signal argument based on $\hat{p}(1-\hat{p})$ and a drift-corrected target that explains saturation. The assumptions, derivations, and empirical saturation evidence are provided in Section 4 and Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper specifies model backbones, training datasets, evaluation benchmarks metrics, OOD protocol, baseline settings, training recipe, SFT and RL hyperparameters, baselines, reward design, editing interfaces, rollout settings, hyperparameters, and prompt templates. These details provide a reasonable path to reproduce or verify the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We plan to publicly release the code, training dataset, prompt templates, and generated data with instructions for data preparation and reproduction of the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper reports the training setup, model backbones, seed data sources, evaluation benchmarks, OOD setting, reward functions, rollout parameters, and inference settings. Additional hyperparameters, implementation details, and prompt templates are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The paper reports deterministic benchmark accuracy results and extensive ablations, but does not include error bars, confidence intervals, or statistical significance tests. We acknowledge this as a limitation and plan to include repeated-run statistics in future versions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification:The paper reports the main hardware setup, including up to 8 A100-80GB GPUs, and provides compute estimates for the Qwen2.5-VL-7B chart-backbone run. Additional training cost and annotation-efficiency comparisons are provided in Appendix G.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public visual reasoning benchmarks and does not involve human subjects, private personal data, or deceptive data collection. The research is intended for benchmark-style VLM reasoning and follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses broader impacts in the Limitations and Impact Statement. On the positive side, VICO can reduce reliance on large-scale human annotation for VLM post-training by turning existing visual data into adaptive, label-valid training samples. On the negative side, automatic image editing may introduce artifacts, label drift, or biased visual changes if not properly verified, and the paper discusses verification and filtering as mitigation strategies.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained models, scraped sensitive datasets, or data intended for identity, surveillance, or safety-critical deployment. The planned release focuses on code, prompts, verifiable editing procedures, and benchmark-style generated training data derived from public visual reasoning datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources for all external datasets, benchmarks, model backbones, and tools used in the experiments, including CLEVR, ChartQA, ZwZ, MathVista, MathVision, MathVerse, WeMath, LogicVista, MMMU, MMVP, BLINK, Qwen, InternVL, SAM3, and prior baseline methods. We respect the licenses and terms of use of these existing assets and use them only for benchmark-style research and evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification:The paper introduces new assets including the VICO training pipeline, EnvRewriter prompts, verifiable editing procedures, generated edited samples, and training trajectories. These assets are documented in the methodology and appendices, including the editing protocols, reward functions, filtering rules, training hyperparameters, implementation details, and prompt templates, and will be released with accompanying documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification:The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper explicitly describes the use of large vision-language models in the core methodology: the Actor and EnvRewriter are initialized from VLM backbones and are jointly trained in the co-evolution loop. The paper also describes the use of GPT-5 and Gemini (Nano Banana) as stronger offline augmentation baselines, and reports the model backbones, prompts, reward functions, and training settings used in the experiments.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.