



VERIEVOL: Scaling Multimodal Mathematical Reasoning via Verifiable Evol-Instruct

Haoling Li^{1,2,*}, Kai Zheng^{2,*}, Jie Wu¹,
Can Xu^{2,†}, Qingfeng Sun², Han Hu², Yujiu Yang^{1,†}

¹Tsinghua University, ²Tencent Hunyuan
*Equal contribution, †Corresponding authors

Abstract

Scaling reinforcement learning for visual mathematical reasoning requires more than generating harder questions: as data volume grows, the reward labels themselves must remain reliable. Yet existing data pipelines scale supervision while trusting the labeller, and policy-side methods assume the underlying answers are already correct. We instead treat scaling as a verifiable data-construction problem and decouple two axes before any policy update: prompt difficulty, expanded by route-specific evolution operators, and answer reliability, enforced by offline hypothesis–test falsification. We instantiate this as VERIEVOL, an iterative framework with two extensible components: a type-aware evolution module that rewrites low-difficulty image-question seeds into harder, image-grounded prompts; and HTV-AGENT, a verifier that accepts an answer only after multi-source counter-evidence has failed to refute it. The resulting verified data scales in volume, extends by adding evolution routes or verifier channels, and plugs directly into existing GRPO-style RL recipes. On a five-benchmark visual-math suite, scaling evolved SFT data from 10K to 250K samples raises the mean accuracy from 35.42 to 54.73; then, with backbone, SFT initialization, and GRPO recipe held fixed, VERIEVOL adds a cumulative +3.88 over an un-evolved RL baseline, of which +1.82 comes from evolved prompts and +2.06 from the HTV-AGENT verifier. We release the prompts, data, models, code, and the full verifier trace of every sample, so that downstream work can scale and audit the pipeline rather than only inspect its outputs.

Correspondence: li-hl23@mails.tsinghua.edu.cn, [\[kevinezheng,leocaxu\]@tencent.com](mailto:[kevinezheng,leocaxu]@tencent.com), yang.yujiu@sz.tsinghua.edu.cn

Website: <https://robertmarton.github.io/verievol>

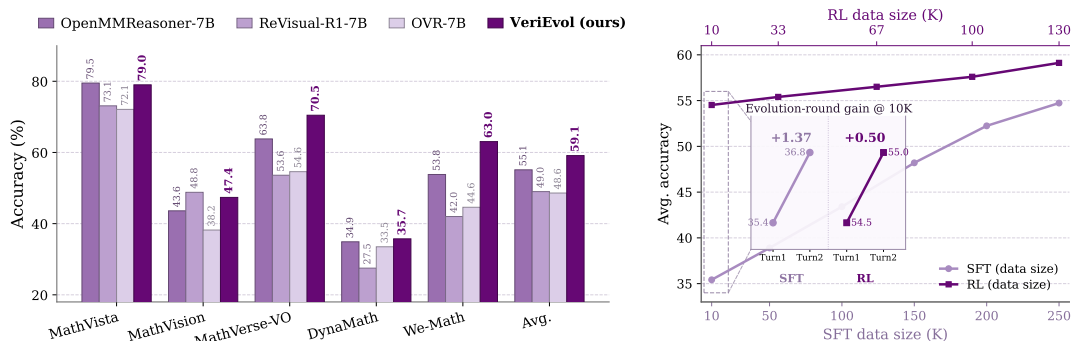


Figure 1 External comparison (left) and data-volume scaling (right). VERIEVOL leads three of five benchmarks and the mean against the strongest external 7B baselines; evolved SFT data (10→250K) and verified RL data (10→130K) both scale monotonically. Details in §4.2–§4.3.

1 Introduction

Reasoning-centric progress in large language models, exemplified by RL-based systems like DeepSeek-R1 [2], has made data quality and reward quality first-order training bottlenecks. The issue is sharper for visual reasoning, where a model must ground variables in diagrams, read values from charts, align symbols with visual structures, and execute multi-step mathematical inference. Existing benchmarks [4–7, 11] show that current multimodal large language models (MLLMs) still fall short on these compositional skills.

Scaling this training regime is not only a matter of collecting more prompts. Many multimodal question-answer pairs are image-grounded but of low difficulty, emphasizing recognition or local extraction over multi-step reasoning; generic synthetic pipelines can increase corpus size while producing questions that are merely longer, weakly visual, or answerable from text priors. The bottleneck becomes more severe at the RL stage: as the number of rollouts and training samples grows, any unverified label is repeatedly converted into reward signal, so label noise is no longer a static annotation error but a training objective that can systematically reinforce incorrect policies.

Two complementary lines of work address parts of this problem. Data-pipeline work [20, 49, 52] scales supervision while trusting the labeller, leaving the answer set unverified; policy-side work [24, 27, 28, 51] reshapes the supervision distribution or modifies GRPO updates while assuming the underlying answers are correct. Both families leave the answer set’s reliability to be fixed **inside** the policy/training loop. We instead make it a property of the data: prompt difficulty scales through route-specific evolution, and—what neither family does explicitly—every answer must survive offline, refutation-seeking hypothesis–test falsification **before** it ever becomes a reward, with this verification decoupled from the policy optimizer. Evolving prompts is shared with prior data-pipeline work; the offline answer-falsification gate is the part that is new. We instantiate this design as VERIEVOL, an iterative framework with two extensible components: type-aware prompt evolution—which routes each seed by problem family (e.g. geometry, charts, OCR) to family-specific operators (§3.2)—and HTV-AGENT, a hypothesis–test verifier that accepts an answer only after multi-source counter-evidence has failed to refute it (§3.3). Because the output is a standard verified prompt–answer–reward tuple, the resulting data can be expanded without changing existing GRPO-style RL recipes.

Our contributions are threefold. **(i)** A scalable multimodal evol-instruct framework that transforms low-difficulty image-question seeds into harder, image-grounded prompts usable for both SFT and RL; in our experiments, increasing SFT data from 10K to 250K raises the mean by +19.31; at the RL stage, scaling verified data from 10K to 130K adds a smooth +4.60 along the **data-volume** axis, while at a fixed 130K budget the **design** itself contributes +3.88 over an un-evolved RL baseline (decomposed as +1.82 from evolved prompts and +2.06 from the verifier). **(ii)** HTV-AGENT, a modular hypothesis–test agent-as-a-judge that screens each answer through four stages—independent solver hypotheses; refutation-seeking verification across counter-evidence, programmatic, and visual channels; conflict resolution by a decider; and a deterministic acceptance gate—so that new verifier channels or answer contracts can be added without re-engineering the policy optimizer. **(iii)** A traceable open release for extensible data construction: in addition to checkpoints and prompts, every training sample is released with its full **verifier trace** (solver hypotheses, counter-evidence reports, tool outputs, decider rationale, per-component gate outcomes), so that downstream work can audit, extend, and rescale construction decisions rather than only inspect final outputs.

2 Related Work

Benchmarks and training data for multimodal reasoning. Canonical benchmarks such as MathVista, MathVerse, MMMU, and MATH-Vision [4–7] serve as the principal evaluation anchors for multimodal reasoning. They are complemented by OlympiadBench [8], DynaMath [9], We-Math [10], and multi-visual math [11], which probe olympiad-level difficulty, input perturbation, knowledge structure, and multi-image composition, respectively; multilingual coverage is further probed by M3Kang [12]. On the training side, Qwen2.5-VL [13], MAMmoTH-VL [17], VisualWebInstruct [18], and MathCoder-VL [19] progressively strengthen backbones and

SFT corpora via cross-modal chain-of-thought, web mining, and code-aided rendering. These efforts remain at SFT scale, however, and do not provide answer verification of RL-grade reliability—i.e., reliable enough to be used as a reward signal, where a wrong label is reinforced across many rollouts rather than seen once. Honey-Data-15M [15] and the multimodal reasoning corpus of Lin et al. [16] are adopted in our experimental setting as disjoint seed pools for SFT and RL, respectively. Both supply large-scale, image-grounded STEM questions, but their answers are predominantly of low difficulty and of insufficient reliability for direct RL supervision.

Prompt evolution for reasoning. Prompt-evolution methods address the difficulty gap. Evol-Instruct [1] and MMEvol [20] systematically rewrite seed instructions into more complex variants, and subsequent data-centric efforts extend this paradigm to multimodal training. Representative examples include MathBook-7B [46], which constructs knowledge-tree-grounded curricula, and OpenMMReasoner [52], which combines 874K evolution-style SFT samples with GSPO-based RL. By contrast, VERIEVOL jointly performs instruction evolution and explicit answer-level falsification, and produces both SFT and RL data within a single, fully traceable construction pipeline.

RL training: reward design, policy optimization, and answer verification. The reliability of the reward signal is a central concern across recent reasoning systems, ranging from text-domain RL such as DeepSeek-R1 and Kimi k1.5 [2, 3] to multimodal judges [21, 22]. We group prior efforts into two families. The first family makes the signal more reliable **from inside the RL loop**, along three complementary directions: **(a) reward design and supervision shape**—visual reinforcement fine-tuning [23] and rule-based rewards on rendered geometry and chart tasks [27, 41] target answer-extraction correctness, while cold-start reasoning traces [24, 25, 35] and self-reflection chains [28] reshape the supervision distribution before or alongside RL; a parallel thread shapes the **visual** signal itself, via perception-anchored attention supervision [38, 40], image-interactive rollouts that “think with images” [48], and trajectory-aware intrinsic reasoning supervision for self-evolving models [34]; **(b) policy optimization** for vision-language models—Skywork-R1V2 [29], OVR [30], DUPL [31], PerL-VL [32], VGPO [37], and Infi-MMR [26] modify GRPO-based updates through reward shaping, advantage normalization, length penalties, or curriculum scheduling; and **(c) data-side sampling**—MMR1 [49] reweights rollouts by variance to stabilize GRPO, and NoisyRollout [47] augments rollout trajectories with controlled visual perturbations. The second family instead equips models with **explicit verification**: at inference time, Self-Refine [53], Reflexion [54], SelfCheckGPT [55], and CRITIC [56] revise drafts via self-feedback or external tools; at training time, PRM800K [57], Math-Shepherd [58], and ReST-EM [59] construct reward-grade supervision via human step-level labels, automatic per-step verification, or self-generated-then-filtered EM loops.

A common assumption across both families is that the underlying answer set is either correct or correctable from inside the policy/training loop; VERIEVOL is orthogonal: it performs answer verification at the data-construction stage (before the policy update), targeting visual-reasoning failure modes that step-level text verifiers cannot diagnose, specifically the visual grounding failures (VG) and text-only shortcuts (TS) we taxonomize in Section 4.4. VERIEVOL shares with the inference-time refinement work the principle that a candidate answer is accepted only after attempts to refute it with counter-evidence have failed, and with the training-time text-domain work the practice of performing verification offline; the resulting VERIEVOL-RL is directly compatible with existing policy-optimization and sampling techniques without modification.

3 VERIEVOL: Evolving Prompts, Falsifying Answers

VERIEVOL takes low-difficulty image-question seeds and produces training samples whose prompts are harder, whose answers have been independently verified, and whose construction decisions are fully traceable. The framework is organized around a scaling principle: separate the components that should grow along different axes. Prompt difficulty scales by adding routes and evolution operators; answer reliability scales by adding independent refutation channels and deterministic gates; training scale grows by materializing the accepted outputs as standard prompt-answer-reward tuples for existing SFT and GRPO recipes. This

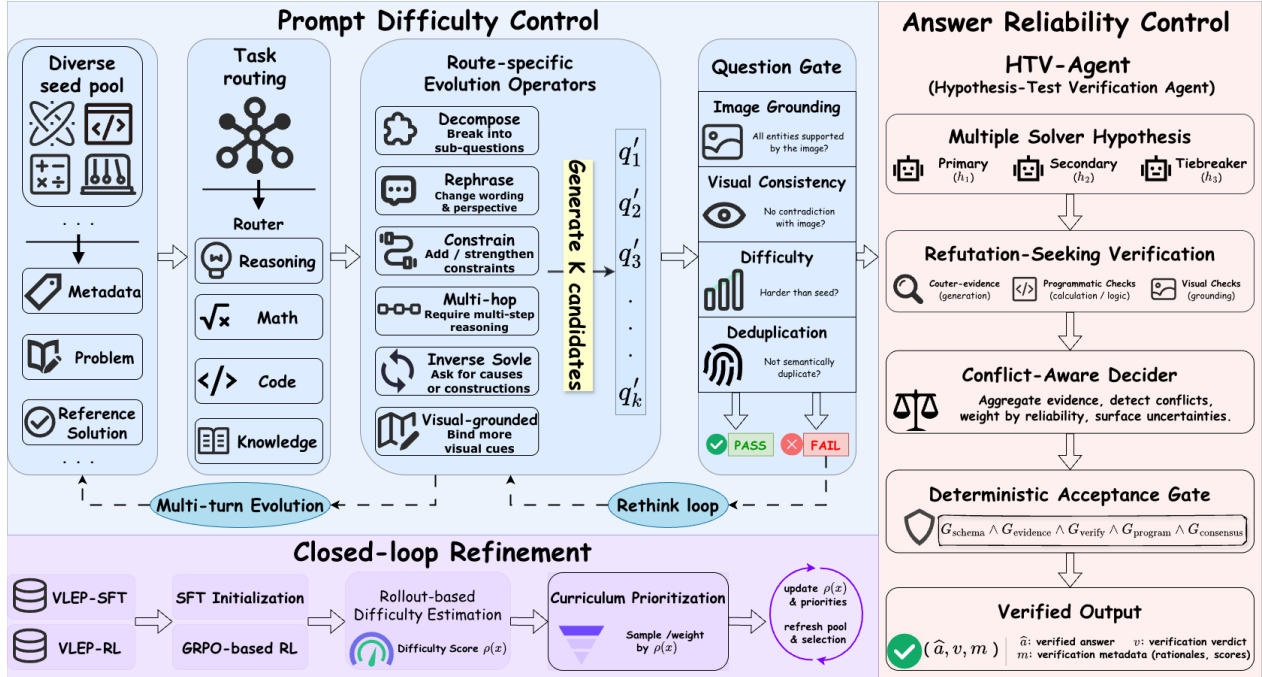


Figure 2 VERIEVOL framework. Two core components turn low-difficulty image-question seeds into verified training samples inside a closed loop. (Top-left, blue) **Prompt Difficulty Control** evolves seeds and filters candidates through a question gate. (Right, orange) **Answer Reliability Control (HTV-AGENT)** falsifies candidate answers and admits them through a deterministic gate. (Bottom-left, purple) **Closed-loop Refinement** materializes accepted samples into VERIEVOL-SFT/VERIEVOL-RL and recycles them into the seed pool. Stages detailed in §3.2, §3.3, and §3.4.

separation avoids a single end-to-end generator whose difficulty, verification, and reward interfaces fail together, and makes each component separately auditable as the data pool expands. Figure 2 shows the two core components, the closed-loop deployment, and their interfaces. §3.1 defines the data objects; §3.2 describes how route-specific operators rewrite seed prompts; §3.3 introduces HTV-AGENT, the hypothesis–test verifier, and its deterministic acceptance gate; and §3.4 specifies how verified samples are materialized into SFT and RL training signal.

3.1 Problem Formulation

A seed sample is denoted $x=(I, q, a)$, where I is an image, q is a seed question, and a is an optional seed answer. Our objective is to construct a training sample $\tilde{x}=(I, q', \hat{a}, m)$, where q' is a more challenging yet still image-grounded prompt, \hat{a} is a verified answer, and m records routing tags, evidence, tool traces, verifier outputs, acceptance decisions, reward-checker specifications, and rollout statistics. The framework returns two data products, $\mathcal{D}_{\text{SFT}}=\{(I, q', r, \hat{a}, e)\}$ and $\mathcal{D}_{\text{RL}}=\{(I, q', \hat{a}, v, m)\}$, where r is a reasoning trace, e is supporting evidence, and v is the verification verdict supporting \hat{a} . A sample can be useful for SFT without being admitted to VERIEVOL-RL; the latter requires stronger answer normalization and deterministic reward-checker guarantees. The full pseudocode of the iterative construction algorithm is provided in Appendix F.

Two design axes are varied independently. **Prompt difficulty** is controlled solely by route-specific operators \mathcal{O}_z acting on the question, $q \rightarrow q'$, while leaving the answer channel untouched; **answer reliability** is controlled solely by the verifier v acting on the candidate answer \hat{a} , independently of how q' was produced. This separation is what lets the two axes be scaled and ablated on their own: empirically, their RL-stage contributions are near-additive (Section 4.2), which is the evidence that they act on orthogonal failure modes rather than a shared one.

3.2 Type-Aware Prompt Evolution

Routing. A single evolution template applied uniformly to all images either over-specifies (e.g., a geometry-style rewrite applied to a chart) or under-specifies (e.g., a generic "make harder" applied to OCR). To match the operator to the image content, we route every sample into a structured tag tuple $z = (t, y, u, c)$, where t is a coarse problem family, y is the answer type, u is a recommended tool set, and c is a compact visual-evidence sketch. The framework figure shows coarse high-level routing categories for readability; the complete visual-route-conditioned taxonomy used to define the operator space is given in Appendix C. The router is implemented as a schema-constrained classifier rather than unconstrained captioning; low-confidence routes fall back to a conservative generic math route or are regenerated with an auxiliary image caption.

Operators. The operator is the unit that produces the harder prompt; we make it route-specific so that the reasoning-skill amplification, the answer format, and the visual-evidence requirement are all tied to the routed tags rather than chosen at generation time. For each route z , VERIEVOL samples an evolution operator from a route-specific set \mathcal{O}_z . Each operator is a constrained prompt template specifying (i) the reasoning skill to amplify, (ii) the visual evidence that must remain necessary, (iii) the desired answer format, and (iv) generator failure modes to avoid. Given K candidates, we score each by image dependence, mathematical lift over q , answerability, verifiability, and a penalty for prompts that turn out to be solvable from text alone; the top candidates are passed to question-level filtering subject to diversity constraints. Figure 5 in Appendix E shows representative before/after pairs, illustrating how a single-step recognition seed is rewritten into a harder, image-grounded reasoning prompt with a short, verifiable answer.

Question-level gate. A generator that scores its own outputs will systematically reward whatever style it produces, so we filter generated prompts with a gate that is independent of the generator: $\phi_q(I, q, q') = \mathbf{1}[C_{\text{img}} \wedge C_{\text{cons}} \wedge C_{\text{vis}} \wedge C_{\text{hard}} \wedge C_{\text{dedup}}]$. The five checks require that q' is grounded in visible image evidence, is visually consistent with the image, fails a text-only probe so that the image remains necessary, introduces a non-trivial difficulty lift over q , and is not a near-duplicate of existing accepted prompts. Schema fields, full operator taxonomy, per-check thresholds, and scoring weights are given in Appendix C.

3.3 HTV-AGENT: Hypothesis-Test Answer Verification

After a prompt passes ϕ_q , HTV-AGENT takes the prompt through four stages: hypothesis generation by independent solvers, refutation-seeking verification with counter-evidence, programmatic, and visual channels, conflict resolution by a decider, and a deterministic acceptance gate. Two design choices are central throughout. **Multiple independent solvers** expose disagreement that a single-solver pipeline cannot, supplying the candidates a verifier needs to compare. **Verifiers asked to refute rather than confirm** probe failure modes a solver has structural reasons to overlook (visual grounding failures, text-only shortcuts, arithmetic slips, in the taxonomy of Section 4.4). The remaining components, the decider and the gate, consume these signals; they do not generate new evidence. Each first answer is therefore treated as a falsifiable hypothesis, never as ground truth. The four stages and the deterministic gate are depicted as the orange "Answer Reliability Control" panel of Figure 2; the loop in the framework is the prompt-rewrite loop used before answer verification when a generated question fails ϕ_q .

Solvers. We run three solver branches at sampling temperatures $\{0.2, 0.6, 0.15\}$: a low-temperature primary solver ($T=0.2$), a higher-temperature secondary solver ($T=0.6$) to encourage independent reasoning trajectories, and a low-temperature tiebreaker ($T=0.15$) that is invoked only when the first two disagree. Each branch returns a tuple $h_b = (a_b, r_b, e_b, \kappa_b)$, where a_b is the candidate answer, r_b the reasoning trace, e_b the claimed evidence, and κ_b a self-reported confidence after normalization.

Verifiers: counter-evidence with programmatic and visual checks. For each candidate cluster, a verifier is prompted to actively look for counter-evidence rather than to confirm the answer: it performs blind elimination on multiple-choice prompts, and back-substitutes the candidate answer into the prompt constraints to search for

contradictions on numerical or symbolic prompts. Two complementary evidence sources back the verifier. A code-checking module translates the relevant arithmetic, algebraic, or combinatorial constraints into Python assertions executed under a restricted interpreter and an AST-safe arithmetic evaluator. A visual-evidence module checks that the cited quantities and symbols are grounded in I via OCR, local crops, and pixel-level structural probes that return measurements such as bounding-box statistics or row/column intensity profiles. Programmatic checks and visual checks are kept as distinct evidence channels because each catches errors the other misses; the evidence-channel taxonomy is summarized in Figure 2, and the concrete tool list is detailed in Appendix C.

Conflict-aware decider. The decider receives the solver hypotheses together with verifier reports, programmatic results, and visual evidence, and decides whether each objection is logically valid and evidence-grounded. It does not assume that the verifier is more reliable than the solver: when verification refutes the solver with sufficient evidence, the answer is revised; when the verifier is inconsistent or unsupported, the solver answer is retained; when neither side is reliable, the sample is rejected rather than forced into the dataset.

Concretely, the decider is a constrained LLM call (no tool use, no code execution) whose prompt asks the model to (i) summarize the verifier’s conclusion in one or two sentences, (ii) assess the **quality** of the verification (logical soundness, self-contradictions, arithmetic errors, unsupported claims) rather than only its conclusion, (iii) commit to a final answer under the three rules above and emit it inside a `<final_answer>` tag, (iv) emit an integer confidence in $[0, 100]$, and (v) optionally emit `<require_rethink>true</require_rethink>` as an abstention flag. Such low-confidence conflicts are treated as non-admissions by the downstream gate. The verbatim decider prompt is reproduced in Appendix C.4.

Deterministic acceptance gate. Candidate answers are first normalized into a canonical form (resolving percentage/decimal equivalence, option letter vs. option text, simple symbolic equivalents, and unit-bearing numbers) and then passed through a deterministic answer gate

$$g(\hat{a}, m) = \mathbf{1}[G_{\text{schema}} \wedge G_{\text{evidence}} \wedge G_{\text{verify}} \wedge G_{\text{program}} \wedge G_{\text{consensus}}], \quad (1)$$

whose components check, respectively, output format (G_{schema}), visual support (G_{evidence}), verifier approval (G_{verify}), executable assertions (G_{program}), and solver-cluster agreement ($G_{\text{consensus}}$). The gate is conjunctive by design: any single failing channel rejects the sample, so admission requires agreement from all evidence sources rather than majority voting.

3.4 From Verified Answers to Training Signal

Materialization for SFT and RL. Accepted samples are materialized differently depending on how they will be used. For VERIEVOL-SFT, we store (I, q', r, \hat{a}, e) : the supervised target is a complete response, so the reasoning trace r and supporting evidence e are retained as imitation signal together with the verified final answer. For VERIEVOL-RL, we store (I, q', \hat{a}, v, m) : the policy must generate its own reasoning during rollout, so the supervised trace is removed, v records the verification verdict supporting \hat{a} , and m carries the canonical answer form and reward-checker specification. This makes the admission bar for VERIEVOL-RL stricter than for VERIEVOL-SFT: an SFT example may be useful when its verified rationale and evidence are high quality, whereas an RL example must additionally admit a deterministic checker such as exact match, numeric tolerance, option matching, symbolic equivalence, or executable assertions.

Reward and rollout-based curriculum. Let ν_m denote the deterministic reward checker reconstructed from the metadata m . The simplest RL reward is $R(y \mid I, q') = \mathbf{1}[\nu_m(y) = \hat{a}] - \beta \mathbf{1}[\text{format_error}(y)]$; metadata also supports richer features (verifier confidence, program-assertion status, cluster size). Finally, after training a student checkpoint we estimate $\rho(x) = \frac{1}{N} |\{i \in [N] : \nu_m(y_i) = \hat{a}\}|$ over N rollouts; examples with intermediate ρ are prioritized for subsequent RL, while $\rho=1$ items are dropped as trivially solvable and $\rho=0$ items as intractable or noisy. The updated $\rho(x)$ values are fed back into seed-pool selection, so later evolution rounds focus on prompts that are neither already solved nor too noisy to verify.

4 Experiments

The experiments evaluate both the effectiveness of VERIEVOL and the scaling properties that motivate the framework: (i) at fixed backbone and optimizer, evolved prompts and verified answers should improve SFT and RL supervision over seed or unverified data (§4.2); (ii) supervision should remain useful as SFT data grows from 10K to 250K samples and verified RL data grows from 10K to 130K samples (§4.3); (iii) the verifier should behave as a modular reliability layer, with complementary evidence channels and residual errors shifting away from visual-grounding failures (§4.3, §4.4).

4.1 Experimental Setup

Models, data, and training. The student is Qwen2.5-VL-7B-Instruct [13] throughout; the proprietary models used inside VERIEVOL (Seed-2.0-Pro for the SFT branch, Gemini-3-flash for the RL branch) act only as data-construction teachers and are never deployed. All RL runs start from a shared checkpoint **VeriEvol-SFT-Init**, obtained by SFT on the 250K-sample VERIEVOL-SFT (STEM seeds from Honey-Data-15M [15]); the SFT data-volume sweep trains the same backbone on smaller VERIEVOL-SFT subsets before any RL. RL is then GRPO on top of this init with a binary correctness reward (Appendix A for the full recipe), using the 130K-sample VERIEVOL-RL pool whose seeds come from the multimodal reasoning corpus of Lin et al. [16] and are restricted to programmatically verifiable answers.

Three RL pipelines and bound argument. We compare three pipelines that differ **only** in the RL data: **RL-Origin** (un-evolved seed prompts), **RL-Evol** (VERIEVOL-evolved prompts, model answers as supervision), and **RL-Evol+Verifier** (the full VERIEVOL, with HTV-AGENT-verified answers). All three share VeriEvol-SFT-Init and the GRPO recipe, so the RL-stage deltas directly isolate the **marginal** effect of evolution and verification when applied at the RL stage; the SFT-stage contribution of evolution is reported separately by the SFT rows (+2.93 over Seed-only SFT) and is not part of the +3.88 RL-stage delta. The remaining confound is an interaction between the evolved init and RL-stage data construction (an evolved init may be more or less responsive to evolved RL data than a seed init); we discuss this residual in Section 5 and do not claim a fully independent decomposition.

Baselines, benchmarks, and protocol. We compare against recent 7B-size visual reasoning baselines: OpenMMReasoner [52], OVR [30], ReVisual-R1 [50], MMR1 [49], WeThink [45], VLAA-Thinker [44], VL-Rethinker [28], ThinkLite-VL [43], MM-Eureka [27], OpenVLThinker [42], TRON [39], Vision-R1 [24], PaLMR [36], NoisyRollout [47], V-Zero [33], and the Qwen2.5-VL-7B-Instruct [13] and InternVL3-8B [14] backbones. Their scores are taken from the original papers or benchmark-complete comparison tables (not re-run), with one exception: OpenMMReasoner-7B’s We-Math-Strict score, which we reproduce under our harness because its paper reports only the loose metric. We evaluate on five held-out benchmarks (MathVista_MINI [4], MathVision_MINI, MathVerse Vision-Only [5], DynaMath-Worst [9], and We-Math-Strict [10]) using each benchmark’s official metric under VLMEvalKit ($T=0.6$, mean over three runs, cross-run $\sigma < 0.5$ on every benchmark). For brevity we use the short names MathVista, MathVision, MathVerse-VO, DynaMath, and We-Math in prose, while tables carry the explicit split/metric suffixes (_MINI, Vision-Only/VO, -Worst, -Strict); these refer to the same evaluation throughout. We single out MathVerse-VO because it most directly tests image reliance. Protocol markers (†) for external entries and the within-row Avg. definition are documented in the caption of Table 1.

4.2 Main Results

Table 1 answers two internal questions and places the answers in external context. Internally, the SFT rows ask whether evolution improves SFT-stage data, and the RL rows ask whether evolution and the verifier each add a separable RL-stage gain on a shared init. The external baseline rows place the resulting numbers against 7B-size baselines.

Method	MathVista Mini \uparrow	MathVision Mini \uparrow	MathVerse VO \uparrow	DynaMath Worst \uparrow	We-Math Strict \uparrow	Avg. \uparrow
External 7B-size baselines.						
OpenMMReasoner-7B [52]	79.50	43.60	63.80	34.90	53.81	55.12
ReVisual-R1-7B [50]	73.10	48.80	53.60	27.50	42.00	49.00
OVR-7B [30]	72.10	38.20	54.60	33.50	44.60	48.60
MMR1-Math-v0 [49]	72.00	29.00	55.40	27.90	31.90	43.24
WeThink-7B [45]	70.90	27.20	44.70	24.40	48.00	43.04
VLAA-Thinker-7B [44]	68.00	26.40	48.20	22.40	41.50	41.30
VL-Rethinker-7B [28]	73.70	28.40	46.40	17.80	36.30	40.52
ThinkLite-VL-7B [43]	71.60	24.60	42.90	16.50	41.80	39.48
MM-Eureka-Qwen-7B [27]	72.60	32.10	45.40	23.00	21.80	38.98
InternVL3-8B [14]	70.50	28.60	33.90	23.00	37.50	38.70
OpenVLThinker-7B [42]	65.30	26.90	38.10	16.80	35.20	36.46
Qwen2.5-VL-7B-Instruct [13]	69.20	21.80	34.10	18.00	32.30	35.08
Supervised Fine-Tuning (ours).						
Seed-only SFT	74.10	36.16	65.02	26.96	56.76	51.80
VERIEVOL-SFT (VeriEvol-SFT-Init)	76.60	39.80	67.01	30.94	59.33	54.73
Reinforcement Learning (ours).						
RL-Origin	77.00	41.45	69.04	30.54	58.19	55.24
RL-Evol	78.10	45.39	69.67	32.14	60.00	57.06
RL-Evol + Verifier (full VERIEVOL)	79.00	47.37	70.46	35.73	63.05	59.12

Table 1 Main results on five visual math reasoning benchmarks. External rows are paper-reported or comparison-table numbers; ours rows (SFT and RL) train Qwen2.5-VL-7B-Instruct under our harness (§4.1). **Bold** marks the per-column best. Blank cells mark benchmarks a paper does not report under our metric; rows covering $\leq 3/5$ benchmarks are shaded lighter and excluded from the Avg.-column bold rule. \uparrow marks a MathVerse split not confirmed as Vision-Only (excluded from the MathVerse-VO column).

SFT-stage evidence. VERIEVOL-SFT (VeriEvol-SFT-Init) lifts the five-benchmark mean from 51.80 to 54.73 over a Seed-only SFT baseline (+2.93), with consistent per-benchmark gains and a peak of +3.98 on DynaMath. Because the VERIEVOL-SFT row differs from the seed baseline only in whether the SFT corpus is evolved, this delta is a clean isolation of the SFT-stage contribution of evolution and confirms that the operators add difficulty that survives SFT before any RL is applied.

RL-stage evidence. On top of VeriEvol-SFT-Init, data construction yields a clean stepwise improvement on every benchmark. Evolution alone (RL-Origin \rightarrow RL-Evol) raises the average from 55.24 to 57.06, peaking on MathVision_MINI (+3.94) and We-Math (+1.81). Adding the HTV-AGENT verifier (RL-Evol \rightarrow RL-Evol + Verifier) adds a further +2.06 on the same average and lifts every benchmark, peaking on DynaMath (+3.59) and We-Math (+3.05). Cumulatively, full VERIEVOL improves over RL-Origin by +3.88 on average and +5.92 on MathVision_MINI. Because all three RL rows share VeriEvol-SFT-Init and the GRPO recipe, these deltas isolate the **marginal** contribution of evolved prompts and the verifier check at the RL stage; the SFT-stage component is bounded by the SFT rows (Section 4.1).

Training dynamics: why evolved prompts help. Figure 3 traces the GRPO reward and policy entropy for the RL-Origin and RL-Evol runs, which differ only in whether the prompt pool is evolved, and explains the +1.82 RL-stage gain mechanistically rather than only as an endpoint accuracy. Two effects are visible. First, on the reward axis the evolved-prompt run climbs faster early and settles at a markedly higher terminal reward (running mean ≈ 0.43 vs. ≈ 0.35 for original prompts), indicating that the evolution operators raise difficulty into a band that still yields a dense, learnable advantage signal rather than saturating the policy. Second, and

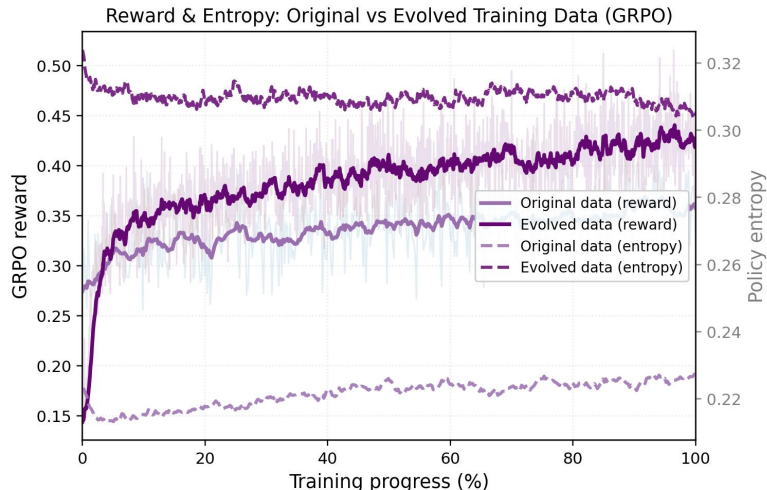


Figure 3 GRPO training dynamics on original vs. evolved prompts. Solid curves are the GRPO reward (left axis); dashed curves are the policy entropy (right axis); the x -axis is normalized training progress. Light traces are the raw per-step signal, bold traces the running mean. Both runs share VeriEvol-SFT-Init, the GRPO recipe, and the verifier, differing only in whether the RL prompt pool is evolved (RL-Evol) or not (RL-Origin).

more diagnostically, the evolved-prompt run **sustains** a higher policy entropy throughout training (≈ 0.31 , roughly flat) whereas the original-prompt run collapses to and then only slowly recovers from a much lower entropy (≈ 0.22). Low-difficulty seeds are solved by a narrow set of trajectories, so GRPO drives the policy toward a low-entropy mode early; the harder, image-grounded evolved prompts keep multiple reasoning routes viable and thus preserve exploration as reward rises. The combination—higher reward without entropy collapse—is the training-time signature of the evolved data and is consistent with RL-Evol’s broad, non-degrading per-benchmark gains.

External comparison: leading on MathVerse-VO. On the four conventional splits VERIEVOL is competitive but not uniformly best (detailed below); we therefore anchor the headline external comparison on MathVerse-VO, the only split that directly isolates image reliance and thus the most diagnostic test of what VERIEVOL targets. Among external baselines, OpenMMReasoner-7B reports the strongest MathVerse-VO score (63.80), followed by MMR1-Math-v0 (55.40) and OVR-7B (54.60); against these competitors, our 70.46 leads by +6.66, +15.06, and +15.86 points respectively.

External comparison: competitive but not uniformly leading on the other four benchmarks. OpenMMReasoner-7B reaches 79.50 on MathVista_MINI (vs. our 79.00) using a roughly $3.5\times$ larger SFT corpus (874K vs. our 250K), and ReVisual-R1-7B reaches 48.80 on MathVision_MINI (vs. our 47.37). On DynaMath-Worst, full VERIEVOL (35.73) leads every full-coverage external baseline (next-best OpenMMReasoner-7B, 34.90). Under the unified We-Math-Strict column, full VERIEVOL reaches 63.05 and leads the strongest external strict score (OpenMMReasoner-7B, 53.81, which we reproduce under We-Math-Strict since its paper reports only the loose metric).

Cross-row averages in the Avg. column should be interpreted with care: some external systems report only a subset of the five benchmarks while our rows cover all five, so a higher external average can reflect averaging over a different subset.

4.3 Scaling Analysis and Ablations

We separate the scaling behavior of VERIEVOL into data volume at the SFT stage, verified data volume at the RL stage, verifier modularity, and evolution depth at a fixed budget.

# SFT data	MathVista Mini ↑	MathVision Mini ↑	MathVerse VO ↑	DynaMath ↑	We-Math ↑	Avg. ↑
10K	66.00	20.39	34.58	19.05	37.06	35.42
50K	68.80	23.38	38.80	21.05	42.45	38.90
100K	72.20	28.64	44.89	24.90	46.40	43.41
150K	74.30	32.68	54.28	28.00	51.75	48.20
200K	75.30	37.93	62.25	29.43	56.29	52.24
250K (VeriEvol-SFT)	76.60	39.80	67.01	30.94	59.33	54.73

Table 2 Scaling evolved SFT data volume. All rows fine-tune Qwen2.5-VL-7B-Instruct with the same SFT recipe and differ only in the number of VERIEVOL-SFT samples. The 250K row is VeriEvol-SFT-Init from Table 1.

Scaling Axis 1: SFT data volume. Table 2 isolates data volume before any RL update. The five-benchmark mean increases monotonically from 35.42 at 10K to 54.73 at 250K (+19.31), showing that the evolved SFT corpus remains highly useful well beyond the small-data regime and justifying the 250K VeriEvol-SFT-Init used by all RL runs. The gains are largest on benchmarks that require stronger visual-compositional coverage: MathVerse-VO rises by +32.43, We-Math by +22.27, and MathVision_MINI by +19.41. MathVista_MINI and DynaMath improve more moderately (+10.60 and +11.89), with DynaMath rising only modestly between 10K and 50K (+2.00), suggesting that these benchmarks either saturate earlier under SFT or depend less on additional supervised coverage. Thus SFT scaling mainly builds the broad visual-reasoning capability of the initialization, while the verified RL sweep below tests whether reward-aligned training still benefits from additional verified samples on top of that initialization.

# RL data	MathVista Mini ↑	MathVision Mini ↑	MathVerse VO ↑	DynaMath ↑	We-Math ↑	Avg. ↑
10K	76.60	38.49	67.26	31.04	59.19	54.52
33K	78.10	38.16	68.02	31.98	60.76	55.40
67K	77.50	41.45	68.91	33.54	61.14	56.51
100K	78.10	45.08	69.29	34.24	61.29	57.60
130K (full VERIEVOL)	79.00	47.37	70.46	35.73	63.05	59.12

Table 3 Scaling verified RL data volume. All rows share VeriEvol-SFT-Init, the same GRPO recipe, and the end-of-one-epoch checkpoint on the corresponding subset of a single verified evolved-prompt pool; they differ only in sample count. The 130K row is the full RL-Evol + Verifier system from Table 1.

Scaling Axis 2: verified RL data volume. Table 3 studies the second scaling axis by varying only the number of verified RL samples while holding the SFT init, GRPO recipe, verifier model, and one-epoch training budget fixed. The five-benchmark mean improves smoothly from 54.52 at 10K to 59.12 at 130K (+4.60 across an order-of-magnitude scaling), showing that the pipeline does not merely produce a high-quality final pool but continues to supply usable reward signal as the pool grows. The improvement is far from uniform: MathVision_MINI is the most data-hungry (38.49 → 47.37, +8.88), MathVerse-VO and We-Math show smaller positive scaling (+3.20 and +3.86), and MathVista_MINI scales least (76.60 → 79.00, +2.40), which we read as near saturation at the backbone capacity. No benchmark worsens with scale, indicating that the verifier filter remains usable across the entire range rather than introducing a systematic bias on any single benchmark.

Scaling Axis 3: verifier-channel modularity. End-to-end RL training is expensive, so we cannot directly run a verifier on/off ablation at matched RL data budget (Limitation ii.a). We instead probe whether HTV-AGENT

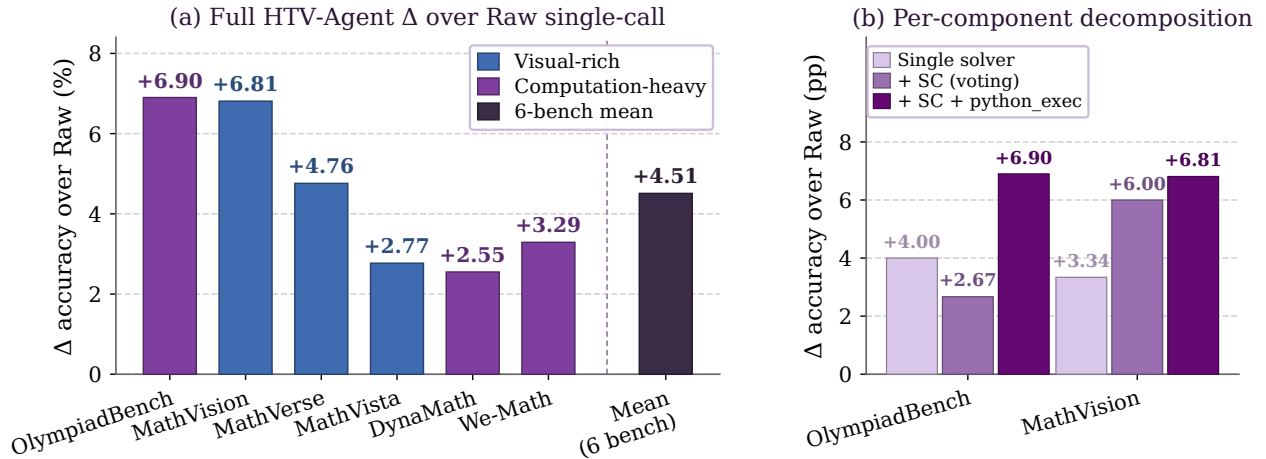


Figure 4 HTV-AGENT component ablation as an inference-time judge (backbone Gemini-3.5-Flash, 300 samples per benchmark). **(a)** Δ accuracy (pp) of the full HTV-AGENT (SC + python_exec) over the **Raw** single-call baseline on the five main benchmarks plus OlympiadBench and their mean; bars are color-coded by benchmark family (visual-rich vs. computation-heavy). **(b)** Per-component decomposition (Raw, Single solver, SC alone, SC + python_exec) on the two benchmarks where all four configurations were run. Full decomposition and latency in Appendix B.

behaves as an extensible reliability layer by deploying it as an **inference-time judge** on six multimodal math benchmarks (the five main benchmarks plus OlympiadBench [8], using Gemini-3.5-Flash as the backbone, 300 samples per benchmark), and ablating its three core components—multi-solver self-consistency, the python_exec tool channel, and the full HTV verifier—by cumulatively enabling them across four configurations: **Raw** (a single LLM call, no agent), **Single Solver** (the agent pipeline with one solver, no self-consistency voting), **SC** (three solvers with self-consistency voting, no python_exec), and **SC + python_exec** (the full HTV-AGENT). Raw is the reference baseline for all reported Δ values; full per-configuration definitions are in Appendix B. This capability-focused ablation uses the stronger Gemini-3.5-Flash backbone, distinct from the more economical Gemini-3-flash teacher used for large-scale VERIEVOL-RL construction in §4.1; the difference is a deliberate cost trade-off (the cheaper model is used at construction scale, the stronger model for this small 300-sample-per-benchmark probe) and the ablation isolates the component design rather than the specific backbone. The reasoning is that if different evidence channels reliably improve raw single-call accuracy in complementary regimes, then the same modular channels are credible as offline filters when constructing verified RL data. Figure 4 reports these configurations.

Three patterns emerge from Figure 4. The **Raw** single-call baseline averages 80.89 over the listed suite (OlympiadBench 63.00, MathVision 83.33, MathVerse 70.33, MathVista 87.67, DynaMath 87.00, We-Math 94.00). First, the full HTV-AGENT (SC + python_exec) improves this mean by +4.51pp and never degrades any benchmark. Second, multi-solver self-consistency is the dominant component on visually rich tasks: MathVision improves by +6.00pp from voting alone, while python_exec adds only +0.81pp on top because the problems are visual rather than computational. Third, python_exec is the dominant component on computation-heavy OlympiadBench (+4.23pp on top of SC), confirming that the programmatic and visual evidence channels in HTV-AGENT are complementary rather than redundant. This complementarity is the key extensibility property: new verifier channels should be added when they target failure modes not covered by existing channels, rather than as undifferentiated extra judging. Because the same evidence channels run inside HTV-AGENT when it screens candidate answers during data construction, these inference-time gains are a credible indirect indicator of HTV-AGENT’s RL-data verification quality, even though they do not substitute for the matched-budget RL ablation that Limitation ii.a documents.

Scaling Axis 4: evolution depth at fixed budget. The data-volume sweeps above vary **how much** data the pipeline uses; we now hold volume fixed and vary **how many evolution rounds** produced it, to test whether deeper

evolution improves data quality independently of scale. Table 4 compares one vs. two rounds of VERIEVOL evolution at a matched 10K budget, under both pure SFT and GRPO RL, on the same five-benchmark suite. Two rounds improve every SFT benchmark (Avg. 35.42 \rightarrow 36.79, +1.37), with the largest gains on MathVista_MINI (+2.42) and MathVision_MINI (+1.57); under RL the mean also improves (54.52 \rightarrow 55.02, +0.50), with all five benchmarks rising and We-Math nearly flat within noise. We read this as evidence that the evolution loop sharpens prompts in a way that is not captured by data volume alone: SFT, which fits the verified labels directly, is more sensitive to prompt quality, while RL at 10K is closer to saturation under our backbone and recipe and benefits more from additional RL volume (Axis 2) than from deeper evolution at fixed volume. Evolution depth and data volume are therefore complementary rather than substitutable.

Regime	Rounds	MathVista Mini \uparrow	MathVision Mini \uparrow	MathVerse VO \uparrow	DynaMath \uparrow	We-Math \uparrow	Avg. \uparrow
SFT	1	66.00	20.39	34.58	19.05	37.06	35.42
	2	68.42	21.96	35.24	20.54	37.80	36.79
RL	1	76.60	38.49	67.26	31.04	59.19	54.52
	2	77.10	39.21	67.54	31.92	59.35	55.02

Table 4 Evolution-depth ablation at a matched 10K data budget. “Rounds” is the number of VERIEVOL evolution passes used to produce the training pool; all other factors (SFT init for the RL rows, GRPO recipe, verifier model, one-epoch budget) are held fixed within each regime. **Bold** marks the per-regime best.

4.4 Error Analysis

The benchmark numbers in Section 4.2 show that VERIEVOL improves accuracy, but they do not by themselves show **why**. To address this, we report a small **pilot** audit on the two benchmarks in our suite that most directly test reliance on the image (MathVerse-VO and MathVision_MINI), and analyze the same questions under RL-Origin and under RL-Evol + Verifier (the full method). The pilot is designed around three questions: (i) does VERIEVOL **redistribute** the model’s errors away from visual grounding failures? (ii) do successful traces actually **use** the image more? (iii) where does the full system still fail? All numbers below come from a 60-sample single-annotator pilot and are illustrative of qualitative trends. The three layers below address (i)–(iii) in turn.

Audit protocol. The pilot draws 60 samples (30 from MathVerse-VO and 30 from MathVision_MINI, no problem family >35% of the pool). We collect both models’ final answer and full chain-of-thought under the same evaluation protocol as Section 4.1 (one trace per problem), and a single annotator labels errors into the five-class taxonomy below. Taxonomy and pilot data are released with the code.

Layer 1: error-type distribution shift. We label each error into five mutually exclusive types: **(VG)** visual grounding, where the model misreads, ignores, or hallucinates an image element; **(RC)** reasoning chain, where visual content is parsed correctly but the multi-step argument breaks; **(SA)** symbolic/arithmetic step error; **(AF)** answer formatting; **(TS)** text-only shortcut. Table 5 reports the joint distribution on the 60-sample pilot. Two trends are visible in the pilot, both reported as absolute counts to keep the small sample size legible. First, VG and TS errors drop from 16 to 9 in absolute terms (out of 28 vs. 24 total errors); their share of the residual error pool also drops from 57% to 38%, suggesting that residual errors shift away from the categories VERIEVOL directly targets. Second, the absolute drop in VG errors (−3 on MathVerse-VO, −4 on MathVision_MINI, −7 in total) is the largest single-category drop in the pilot, in the same direction as the +1.42 MathVerse-VO and +5.92 MathVision_MINI gains in Table 1. RC and SA counts are essentially flat or slightly higher in absolute terms (2 \rightarrow 2, 1 \rightarrow 2 on MathVerse-VO; 4 \rightarrow 5, 3 \rightarrow 4 on MathVision_MINI), so in a smaller error pool they become a **larger share** of the residual; this is the backbone-capacity pattern we

describe under Layer 3 (R2). At $n=60$ with a single annotator we draw no quantitative conclusion from these proportions.

Setting (per benchmark slice)	Errors per category (out of 30, ↓)					Total errors ↓
	VG	RC	SA	AF	TS	
<i>MathVerse-VO (30 audited)</i>						
RL-Origin	5	2	1	1	0	9
RL-Evol + Verifier (full)	2	2	2	1	1	8
<i>MathVision_MINI (30 audited)</i>						
RL-Origin	8	4	3	1	3	19
RL-Evol + Verifier (full)	4	5	4	1	2	16

Table 5 Pilot error-type distribution from a 60-sample audit (30 per benchmark) by a single annotator. VG = visual grounding, RC = reasoning chain, SA = symbolic/arithmetic, AF = answer formatting, TS = text-only shortcut. Counts are illustrative of the qualitative pattern and are not significance-tested.

Layer 2: image use in successful traces. Aggregate error counts are necessary but not sufficient: a system can score higher while still relying on text-only patterns. To check whether the Layer-1 shift is accompanied by qualitatively different traces, we extract the pilot problems on which RL-Origin fails and RL-Evol + Verifier succeeds ($n=5$ on MathVerse-VO; $n=7$ on MathVision_MINI; $n=12$ in total) and score each successful trace, paired against the corresponding failed RL-Origin trace on the same problem, on three binary criteria: explicit image reference, image-grounded subgoal decomposition, and answer consistency with image content (vs. text-only priors). Across the 12 pairs, RL-Evol + Verifier satisfies the first criterion on 10/12 (83%) of its successful traces versus 5/12 (42%) for the matched failed RL-Origin traces, with similar gaps on the other two criteria (9/12 vs. 3/12; 11/12 vs. 5/12). Two case studies in Appendix D illustrate the shift; statistical significance at this n is not claimed.

Layer 3: residual failure modes. VERIEVOL still fails along two patterns observed in the pilot. **(R1) Solver-verifier shared blind spots:** roughly 3 of the 60 audited problems ($\sim 5\%$) where all three solver branches and the verifier converge on the same incorrect visual reading and the gates accept it. These are the cases where the HTV-AGENT "refute-not-confirm" framing (§3.3) reduces to "confirm" because every channel makes the same mistake; a heterogeneous verifier ensemble (different backbone, different tool stack) is the natural remedy. **(R2) Backbone-capacity ceiling:** long-tail MathVision_MINI items requiring multi-step symbolic chains beyond the 7B backbone’s reliable reasoning length, manifesting as RC/SA errors largely insensitive to the data framework (these account for 12 of the 60 audited problems ($\sim 20\%$) and explain the flat-or-rising RC/SA columns in Table 5); a larger backbone is the natural remedy.

5 Limitations, Broader Impact, and Responsible Release

Limitations. VERIEVOL depends on seed-pool coverage: evolution cannot fully compensate for under-represented domains or diagram styles, and the gates, though conservative, are imperfect—evolved prompts may still leak textual cues, and HTV-AGENT can accept a wrong answer when solvers, verifiers, and tools share a blind spot. The RL-ready subset is therefore necessarily narrower than the SFT-ready subset. On the empirical side, each RL run uses a single seed, so single-benchmark differences below ~ 1 point should be read as suggestive rather than significant, and the verifier-on/off and SFT-stage-evolution effects are isolated only indirectly through the SFT and RL rows of Table 1 rather than through fully matched controls. These are scoping choices that bound the granularity of our claims, not threats to the main scaling trends.

Positive impact, risks, and safeguards. The framework can reduce the cost of building high-quality training data for visual math reasoning and make the construction process transparent for smaller teams. Synthetic pipelines can amplify source bias, overfit to generator/judge preferences, or create spurious reward shortcuts;

raw images with restrictive licenses or privacy concerns pose legal/ethical risks. We release per-source license tracking, dedup, and content filtering, and substitute metadata plus source pointers for raw images when redistribution rights are unclear.

6 Conclusion

We framed RL data construction for visual mathematical reasoning as a scaling problem: prompt difficulty must grow without losing image dependence, answer labels must remain reliable as the pool expands, and the resulting supervision must plug into existing policy optimizers. VERIEVOL addresses these requirements by decoupling route-specific prompt evolution from hypothesis–test answer falsification before any policy update.

At the SFT stage, scaling evolved data from 10K to 250K raises the five-benchmark mean by +19.31; with backbone, SFT init, and GRPO recipe held fixed, the RL-stage design further improves the mean by +3.88 over an un-evolved RL baseline (+5.92 on MathVision_MINI) and continues to scale from 10K to 130K verified RL samples (+4.60). The component analysis supports another scaling axis: verifier channels are complementary rather than redundant, so reliability can be extended by adding targeted refutation channels instead of redesigning the optimizer.

The broader takeaway is composability. Data-stage verification is orthogonal to policy-side innovations, allowing future GRPO-style optimizers, length penalties, or rollout reweighting strategies to build on VERIEVOL-RL directly; it also suggests extensions beyond mathematical reasoning to chart QA, diagram-based science, and document understanding, wherever image-grounded answers admit deterministic or programmatic checks. Code, data, models, and verifier traces will be released so that downstream work can audit, extend, and rescale the construction pipeline.

References

- [1] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. WizardLM: Empowering large language models to follow complex instructions. arXiv preprint arXiv:2304.12244, 2023.
- [2] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.
- [3] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, and others. Kimi k1.5: Scaling reinforcement learning with LLMs. arXiv preprint arXiv:2501.12599, 2025.
- [4] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyang Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In Proceedings of ICLR, 2024.
- [5] Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. MathVerse: Does your multi-modal LLM truly see the diagrams in visual math problems? In Proceedings of ECCV, 2024.
- [6] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and others. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In Proceedings of CVPR, 2024.
- [7] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with MATH-Vision dataset. In Advances in Neural Information Processing Systems Datasets and Benchmarks Track, 2024.
- [8] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Proceedings of ACL, 2024.

- [9] Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. DynaMath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In International Conference on Learning Representations (ICLR), 2025.
- [10] Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong, Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. We-Math: Does your large multimodal model achieve human-like mathematical reasoning? arXiv preprint arXiv:2407.01284, 2024.
- [11] Peijie Wang, Zhong-Zhi Li, Fei Yin, Xin Yang, Dekang Ran, and Cheng-Lin Liu. MV-MATH: Evaluating multimodal math reasoning in multi-visual contexts. In Proceedings of CVPR, 2025.
- [12] Aleix Torres-Camps, Nathaniel Mitrani Hadida, Víctor Conchello Vendrell, Àlex Batlle Casellas, Arnau Padrés Masdemont, and Jordi Ros-Giralt. M3Kang: Evaluating multilingual multimodal mathematical reasoning in vision-language models. arXiv preprint arXiv:2601.16218, 2026.
- [13] Qwen Team. Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923, 2025.
- [14] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479, 2025.
- [15] Open-Bee Team. Honey-Data-15M: A large-scale open multimodal instruction-tuning dataset. <https://huggingface.co/datasets/Open-Bee/Honey-Data-15M>, 2025.
- [16] Honglin Lin, Zheng Liu, Yun Zhu, Chonghan Qin, Juekai Lin, Xiaoran Shang, Conghui He, Wentao Zhang, and Lijun Wu. MMFineReason: Closing the multimodal reasoning gap via open data-centric methods. arXiv preprint arXiv:2601.21821, 2026.
- [17] Jarvis Guo, Tuney Zheng, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Yizhi Li, Graham Neubig, Wenhui Chen, and Xiang Yue. MAMmoTH-VL: Eliciting multimodal reasoning with instruction tuning at scale. In Proceedings of ACL, 2025.
- [18] Yiming Jia, Jiachen Li, Xiang Yue, Bo Li, Ping Nie, Kai Zou, and Wenhui Chen. VisualWebInstruct: Scaling up multimodal instruction data through web search. arXiv preprint arXiv:2503.10582, 2025.
- [19] Ke Wang, Junting Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, and Hongsheng Li. MathCoder-VL: Bridging vision and code for enhanced multimodal mathematical reasoning. In Findings of ACL, 2025.
- [20] Run Luo, Haonan Zhang, Longze Chen, Ting-En Lin, Xiong Liu, Yuchuan Wu, Min Yang, Minzheng Wang, Pengpeng Zeng, Lianli Gao, and others. MMEvol: Empowering multimodal large language models with Evol-Instruct. arXiv preprint arXiv:2409.05840, 2024.
- [21] Renjie Pi, Felix Bai, Qibin Chen, Simon Wang, Jiulong Shan, Kieran Liu, and Meng Cao. MR. Judge: Multimodal reasoner as a judge. arXiv preprint arXiv:2505.13403, 2025.
- [22] Shu Pu, Yaochen Wang, Dongping Chen, Yuhang Chen, Guohao Wang, Qi Qin, Zhongyi Zhang, Zhiyuan Zhang, Zetong Zhou, Shuang Gong, and others. Judge Anything: MLLM as a judge across any modality. arXiv preprint arXiv:2503.17489, 2025.
- [23] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-RFT: Visual reinforcement fine-tuning. arXiv preprint arXiv:2503.01785, 2025.
- [24] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-R1: Incentivizing reasoning capability in multimodal large language models. arXiv preprint arXiv:2503.06749, 2025.
- [25] Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, Bo Zhang, and Wei Chen. R1-Onevision: Advancing generalized multimodal reasoning through cross-modal formalization. arXiv preprint arXiv:2503.10615, 2025.
- [26] Zeyu Liu, Yuhang Liu, Guanghao Zhu, Congkai Xie, Zhen Li, Jianbo Yuan, Xinyao Wang, Qing Li, Shing-Chi Cheung, Shengyu Zhang, Fei Wu, and Hongxia Yang. Infi-MMR: Curriculum-based unlocking multimodal reasoning via phased reinforcement learning in multimodal small language models. arXiv preprint arXiv:2505.23091, 2025.

- [27] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, and others. MM-Eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. arXiv preprint arXiv:2503.07365, 2025.
- [28] Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhui Chen. VL-Rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. arXiv preprint arXiv:2504.08837, 2025.
- [29] Peiyu Wang, Yichen Wei, Yi Peng, Xiaokun Wang, Weijie Qiu, Wei Shen, Tianyidan Xie, Jiangbo Pei, Jianhao Zhang, Yunzhuo Hao, Xuchen Song, Yang Liu, and Yahui Zhou. Skywork R1V2: Multimodal hybrid reinforcement learning for reasoning. arXiv preprint arXiv:2504.16656, 2025.
- [30] Yana Wei, Liang Zhao, Jianjian Sun, Kangheng Lin, Jisheng Yin, Jingcheng Hu, Yinmin Zhang, En Yu, Haoran Lv, Zejia Weng, Jia Wang, and others. Open Vision Reasoner: Transferring linguistic cognitive behavior for visual reasoning. arXiv preprint arXiv:2507.05255, 2025.
- [31] Rui Liu, Dian Yu, Tong Zheng, Runpeng Dai, Zongxia Li, Wenhao Yu, Zhenwen Liang, Linfeng Song, Haitao Mi, Pratap Tokekar, and Dong Yu. Dual-uncertainty guided policy learning for multimodal reasoning. arXiv preprint arXiv:2510.01444, 2025.
- [32] Hoang Anh Just, Yifei Fan, Handong Zhao, Jiuxiang Gu, Ruiyi Zhang, Simon Jenni, Kushal Kafle, Ruoxi Jia, and Jing Shi. More than the final answer: Improving visual extraction and logical consistency in vision-language models. arXiv preprint arXiv:2512.12487, 2025.
- [33] Han Wang, Yi Yang, Jingyuan Hu, Minfeng Zhu, and Wei Chen. V-Zero: Self-improving multimodal reasoning with zero annotation. arXiv preprint arXiv:2601.10094, 2026.
- [34] Meghana Sunil, Manikandarajan Venmathimaran, and Muthu Subash Kavitha. iReasoner: Trajectory-aware intrinsic reasoning supervision for self-evolving large multimodal models. In Findings of the Association for Computational Linguistics (ACL), 2026. arXiv:2601.05877.
- [35] Ruilin Luo, Chufan Shi, Yizhen Zhang, Cheng Yang, Songtao Jiang, Tongkun Guan, Ruizhe Chen, Ruihang Chu, Peng Wang, Mingkun Yang, Yujiu Yang, Junyang Lin, and Zhibo Yang. From narrow to panoramic vision: Attention-guided cold-start reshapes multimodal reasoning. In International Conference on Learning Representations (ICLR), 2026. arXiv:2603.03825.
- [36] Yantao Li, Qiang Hui, Chenyang Yan, Kanzhi Cheng, Fang Zhao, Chao Tan, Huanling Gao, Jianbing Zhang, Kai Wang, Xinyu Dai, and Shiguo Lian. PaLMR: Towards faithful visual reasoning via multimodal process alignment. In CVPR Findings, 2026. arXiv:2603.06652.
- [37] Zengbin Wang, Feng Xiong, Liang Lin, Xuecai Hu, Yong Wang, Yanlin Wang, Man Zhang, and Xiangxiang Chu. Visually-guided policy optimization for multimodal reasoning. arXiv preprint arXiv:2604.09349, 2026.
- [38] Ruina Hu, Chen Wang, Lai Wei, Jionghao Bai, Bin Yu, Weiran Huang, Kai Wang, and Yue Wang. Attend to evidence: Evidence-anchored spatial attention supervision for multimodal RLVR. arXiv preprint arXiv:2605.30912, 2026.
- [39] Tianze Yang, Yucheng Shi, Ruitong Sun, Jingyuan Huang, Ninghao Liu, and Jin Sun. TRON: Targeted rule-verifiable online environments for visual reasoning RL. arXiv preprint arXiv:2606.01599, 2026.
- [40] Shuoshuo Zhang, Yizhen Zhang, Jingjing Fu, Lei Song, Jiang Bian, Yujiu Yang, and Rui Wang. See less, see right: Bi-directional perceptual shaping for multimodal reasoning. arXiv preprint arXiv:2512.22120, 2026.
- [41] Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, Vinci, and Zihao Yue. R1-V: Reinforcing super generalization ability in vision-language models with less than three dollars. Technical report, 2025. <https://github.com/StarsfieldAI/R1-V>.
- [42] Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. OpenVLThinker: Complex vision-language reasoning via iterative SFT-RL cycles. arXiv preprint arXiv:2503.17352, 2025.
- [43] Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. ThinkLite-VL: Reasoning-enhanced vision-language models with sample-efficient reinforcement fine-tuning. arXiv preprint arXiv:2504.07934, 2025.

- [44] Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. VLAA-Thinker: SFT or RL? An early investigation into training R1-like reasoning large vision-language models. *Transactions on Machine Learning Research*, 2025.
- [45] Jie Yang, Feipeng Ma, Zitian Wang, Dacheng Yin, Kang Rong, Fengyun Rao, and Ruimao Zhang. WeThink: Toward general-purpose vision-language reasoning via reinforcement learning. *arXiv preprint arXiv:2506.07905*, 2025.
- [46] Runqi Qiao, Qiuna Tan, Peiqing Yang, Yanzi Wang, Xiaowan Wang, Enhui Wan, Sitong Zhou, Guanting Dong, Yuchen Zeng, Yida Xu, Jie Wang, Chong Sun, Chen Li, and Honggang Zhang. We-Math 2.0: A versatile MathBook system for incentivizing visual mathematical reasoning. *arXiv preprint arXiv:2508.10433*, 2025.
- [47] Xiangyan Liu, Jinjie Ni, Zijian Wu, Chao Du, Longxu Dou, Haonan Wang, Tianyu Pang, and Michael Qizhe Shieh. NoisyRollout: Reinforcing visual reasoning with data augmentation. *Advances in Neural Information Processing Systems*, 2025. *arXiv:2504.13055*.
- [48] Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing “thinking with images” via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. *arXiv:2505.14362*.
- [49] Sicong Leng, Jing Wang, Jiayi Li, Hao Zhang, Zhiqiang Hu, Boqiang Zhang, Yuming Jiang, Hang Zhang, Xin Li, Lidong Bing, Deli Zhao, Wei Lu, Yu Rong, Aixin Sun, and Shijian Lu. MMR1: Enhancing multimodal reasoning with variance-aware sampling and open resources. *arXiv preprint arXiv:2509.21268*, 2025.
- [50] Yuhao Chen, Shubin Huang, Hongyi Yu, Long Li, Zihan Wang, Xinyi Wang, Yuwei Yan, Lifan Yuan, Zhihao Bai, Mengmeng Liu, Jiongnan Liu, Mengjie Wang, Wei Tang, Liuxin Zhang, Junlong Wu, Mingsheng Long, Hao Zhao, Jianzhuang Liu, and Yiming Yang. ReVisual-R1: An open-source 7B multimodal large language model for deep reasoning. *arXiv preprint arXiv:2506.04207*, 2025.
- [51] Zhenghai Wang, Wenxuan Zhang, Wenhao Yu, Tianhao Wu, Heng Ji, Hongming Zhang, Dong Yu, Manling Li, and Kaixin Ma. Perception-aware policy optimization for multimodal reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. *arXiv:2507.06448*.
- [52] Kaichen Lin, Bo Li, Yuanhan Zhang, Yifei Sun, Yixiu Liu, Pengyun Wang, Yuhao Dong, Wenjia Liu, Xinyu Wang, Zhiqi Bu, Ziwei Liu, and Chunyuan Li. OpenMMReasoner: Pushing the frontiers of multimodal reasoning with an open and reproducible recipe. *arXiv preprint arXiv:2511.16334*, 2025.
- [53] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.
- [54] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.
- [55] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.
- [56] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024.
- [57] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.
- [58] Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- [59] Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper

Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. Transactions on Machine Learning Research (TMLR), 2024.

A Training Compute and Hyperparameters

This appendix documents the training configuration of VERIEVOL across the SFT and RL stages. Hyperparameter values reported here are consistent with the description in the main text (Section 4.1). We specify the hardware family and the deterministic configuration manifest under which every reported run can be reproduced.

SFT stage (VeriEvol-SFT-Init). We fine-tune Qwen2.5-VL-7B-Instruct on the 250K-sample VERIEVOL-SFT corpus to obtain VeriEvol-SFT-Init. We use sequence packing with a global batch size of 512, AdamW with a constant learning rate of 5×10^{-5} , and train for 10 epochs. The resulting checkpoint serves as the starting point for every RL run reported in the paper.

RL stage (GRPO on VERIEVOL-RL). All RL variants share the GRPO recipe described in Section 4.1: global batch size 256, group size 16, dynamic batching, and a binary reward of 1 when the model’s extracted answer matches the verified reference under the verifier v and 0 otherwise (with a small format-error penalty β). The same prompt format and the same reference-answer extraction routine are used across RL-Origin, RL-Evol, RL-Evol + Verifier, and the four data-scale ablation runs.

Compute and reproducibility. All SFT and RL runs were carried out on a homogeneous H20 (80 GB) GPU cluster, with each run using a single 8-GPU node and tensor-parallelism degree 8; the SFT run additionally uses sequence-packed data parallelism across two such nodes. As an order-of-magnitude indication of the total compute, the reported experiments comprise one 250K-sample SFT run plus 8 GRPO runs (RL-Origin, RL-Evol, RL-Evol+Verifier, the four VERIEVOL-RL scale points at 10/33/67/100/130K with 130K shared, and the round-2 ablation), each on the same 8×H20 node. For exact reproduction we release: (i) a single configuration manifest (YAML) for the SFT and the 130K RL run that pins the framework version, optimizer, scheduler, batch dimensions, group size, KL coefficient, response-length budget, and seed; and (ii) the full VERIEVOL-RL and VERIEVOL-SFT artifacts and verifier traces, so that any third party can rerun the released configurations on equivalent hardware.

B HTV-Agent Component Ablation: Full Details

This appendix expands Figure 4 in the main text. We deploy HTV-AGENT as an inference-time judge on the five main multimodal math benchmarks plus OlympiadBench and ablate its three core components (multi-solver self-consistency, programmatic execution via the `python_exec` tool channel, and the verifier filter that decides whether to accept or fall back to the raw answer). The goal is not to claim a new state-of-the-art at inference time, but to provide indirect evidence that the same component design is a reliable filter when used offline to verify candidate RL training data.

Setup. The backbone for every configuration is Gemini-3.5-Flash, accessed through a shared API gateway. Each benchmark draws 300 samples. Common high-effort settings: `reasoning_effort=high`, `max_tokens=8192`, `max_solver_steps=16`, `max_verifier_steps=10`, per-sample timeout 600 s. Four configurations are compared.

- **Raw Baseline.** A single LLM call, no agent. Direct prompt-to-answer.
- **Single Solver.** The agent pipeline with one solver (no self-consistency voting), with `python_exec` enabled on computation-related benchmarks and the verifier filter active.
- **SC (no `python_exec`).** Three solvers with self-consistency voting, but the `python_exec` tool channel is disabled; the verifier filter is active.
- **SC + `python_exec`.** The full HTV-AGENT pipeline with three solvers, self-consistency voting, `python_exec` tool, and the verifier filter.

Full agent-ablation accuracy table. Table 6 reports the absolute accuracy of each configuration on the five main benchmarks plus OlympiadBench. Figure 4 in the main text plots the Δ -over-Raw view of the same data; this table provides the absolute numbers for reproducibility, including the two configurations (Single Solver and Raw) that were run on the full listed suite and the SC-only configuration that was run only on OlympiadBench and MathVision (see Notes on data validity below).

Benchmark	Raw \uparrow	Single Solver \uparrow	SC (no exec) \uparrow	SC + <code>python_exec</code> \uparrow	Best Δ
OlympiadBench	63.00	67.00	65.67	69.90	+6.90
MathVision	83.33	86.67	89.33	90.14	+6.81
MathVerse	70.33	72.33	–	75.09	+4.76
MathVista	87.67	88.33	–	90.44	+2.77
DynaMath	87.00	87.67	–	89.55	+2.55
We-Math	94.00	96.67	–	97.29	+3.29
Mean (listed suite)	80.89	83.11	–	85.40	+4.51

Table 6 Full HTV-AGENT component ablation as an inference-time judge: absolute accuracies on the five main benchmarks plus OlympiadBench (backbone Gemini-3.5-Flash, 300 samples per benchmark; absolute-value counterpart of Figure 4). “–” marks configurations not run on that benchmark (the SC-only configuration was a targeted probe on OlympiadBench and MathVision; see Notes on data validity). **Bold** marks the per-benchmark best; **Best Δ** reports the gap to Raw. Within-row Mean covers only the benchmarks each row runs.

Per-component decomposition: isolating the role of self-consistency. Table 7 re-examines the two benchmarks where all four configurations were run, with the explicit goal of isolating **how much of the full-pipeline gain is already explained by self-consistency voting alone**. We highlight the SC-only row because it shows that on visually rich MathVision, three-solver voting recovers +6.00pp without any code execution—i.e., SC alone is the dominant component on that benchmark, and `python_exec` adds only +0.81pp further. On computation-heavy OlympiadBench the picture inverts: SC alone only adds +2.67pp, while adding `python_exec` on top of SC delivers an additional +4.23pp, making the programmatic channel the dominant component there. This per-benchmark inversion is the clearest evidence we have that the two evidence channels are complementary rather than redundant.

Configuration	Accuracy (\uparrow) ; (Δ over Raw)		
	OlympiadBench	MathVision	Mean of 2
Raw Baseline	63.00	83.33	73.17
+ Single Solver (no voting)	67.00 (+4.00)	86.67 (+3.34)	76.84 (+3.67)
+ Self-consistency voting (no exec)	65.67 (+2.67)	89.33 (+6.00)	77.50 (+4.33)
+ <code>python_exec</code> on top of SC	69.90 (+6.90)	90.14 (+6.81)	80.02 (+6.86)

Table 7 Per-component decomposition on the two benchmarks where all four configurations were run (300 samples each). **Shaded row**: self-consistency voting alone (without `python_exec`). Values in parentheses are Δ over Raw; **bold** marks the per-column best.

Latency. The full HTV-AGENT pipeline is substantially more expensive than a single LLM call because it runs three solver passes plus the verifier. Table 8 reports per-sample wall-clock cost. This is the price for inference-time gains, but at the data-construction stage the cost is amortized over the lifetime of the verified VERIEVOL-RL pool; it is paid once and reused across every downstream RL run.

Configuration	Latency per sample	Relative to Raw
Raw Baseline	14–35 s	1×
Single Solver	55–97 s	4–6×
SC (no <code>python_exec</code>)	367–433 s	12–25×
SC + <code>python_exec</code>	344–465 s	15–20×

Table 8 Inference-time latency per sample, relative to a single Raw API call. At the data-construction stage this cost is paid once per training sample and reused across every RL run that consumes VERIEVOL-RL.

Key qualitative findings. Three observations summarize the table. (1) The full HTV-AGENT never degrades the listed-suite mean and never drops any single benchmark below the raw baseline by more than the validity caveats listed below; the multi-solver voting plus a verifier filter that can abstain back to the raw answer act as a safety net. (2) The two evidence channels (multi-solver voting versus programmatic execution) are complementary: voting helps most when reasoning paths benefit from diversity, while `python_exec` helps most when problems involve explicit arithmetic. (3) The agent pipeline had zero pipeline-level errors across all five benchmarks in our runs, while the raw baseline occasionally suffered timeout or connection errors; this robustness is a side benefit of the multi-solver retry architecture and is one of the reasons the same architecture is a reliable offline filter for VERIEVOL-RL construction.

Notes on data validity. Because we ran multiple configurations across two seeds and two API channels during the ablation campaign, several caveats apply. **(i)** The SC and `python_exec`-related configurations were run on seed 7, while the v3 single-solver and SC-only ablations were run on seed 42. Cross-seed comparisons are therefore approximate. **(ii)** A small number of SC runs were terminated before completing all 300 samples (e.g., SC on OlympiadBench completed 299/300 because the last sample was killed; SC on MathVision had 213/300 valid pairs after API budget exhaustion; the SC + `python_exec` run on We-Math reports the score over 255/300 valid samples after API budget exhaustion); the table reports each benchmark’s accuracy over the largest available valid subset. **(iii)** Two API channels (`api_google` and `api_naci`) were used; both route to the same underlying Gemini-3.5-Flash model.

C Evolution Operator Design Space and the 12-Topic Instantiation

This appendix presents the design space we considered when prototyping VERIEVOL, organized along four axes: the visual route, the reasoning skill to amplify, the answer type, and the verification contract. We list the full taxonomy here for two reasons. First, the operator design is itself a contribution: it documents the kinds of constraints a multimodal evol-instruct framework has to respect to produce both SFT-usable and RL-usable data, and it serves as a practical checklist for adapting VERIEVOL to new domains. Second, our current implementation is a deliberate restriction of this design space: rather than instantiating every cell of the four-axis cross product, the production pipeline collapses the routing axes into a 12-topic vocabulary (OCR, image description, detection, analysis, content creation, suggestions, summarization, logical reasoning, science, concept extraction, medical imaging, and scene understanding), and associates one evolution prompt template per topic. Each template bundles the relevant visual-context preservation, reasoning-difficulty operator, answer-type constraint, and verifier contract for that topic, rather than composing them at runtime. The taxonomy below should therefore be read as the upper bound of the design we explored, with the 12-topic instantiation being the subset that has been deployed and audited in this paper. We separate operators into the four axes mentioned above; a generation prompt selects one route, applies the route’s reasoning operator, and attaches the route’s answer-type operator with its corresponding verifier.

Table 9 Full visual-route-conditioned evolution operator taxonomy.

Route	Trigger evidence	Evolution operators	Verification / rejection contract
Geometry diagrams	Points, lines, angles, polygons, circles, labels, congruence marks	Auxiliary construction · relation composition · theorem/invariant query · coordinate conversion · inverse solve from target value · ratio, perimeter, area, or length comparison · multi-step equation setup · degenerate-case exclusion	Every symbol must bind to a visible mark · reject impossible measurements, purely text-only theorem recall, and questions requiring precision not supported by the diagram
Coordinate geometry and function plots	Axes, ticks, grids, curves, intercepts, shaded regions, plotted points	Graph-to-equation mapping · intercept/extremum/intersection query · slope or rate computation · area/accumulation approximation · parameter-shift reasoning · symbolic-numeric consistency check · interval selection	Require coordinate evidence · normalize axis scales · reject unseen precision, extrapolation outside visible support, and ambiguous intersections
Statistical charts	Axes, legends, bars, lines, pies, scales, error bars	Cross-series comparison · extrema/ranking · aggregate over visible groups · rate or slope change · interpolation inside visible range · unit/scale/log conversion · conditional trend query · uncertainty-aware comparison	Require cited axis/legend evidence · reject chart-title-only questions, unsupported extrapolation, and ambiguous visual encodings
Tables and spreadsheets	Rows, columns, headers, cells, subtotals, footnotes	Row-column join · multi-cell aggregation · conditional filtering · subtotal or percentage computation · rank/order query · unit normalization across columns · consistency check against totals	Require header-cell alignment · reject hallucinated rows/columns, implicit database facts, and aggregation over hidden cells
OCR and document math	Local text spans, formulas, captions, page layout, footnotes, boxed expressions	Multi-region lookup · formula-text alignment · equation extraction and solving · distractor text suppression · layout-conditioned aggregation · cross-reference between caption, figure, and formula · variable definition lookup	Require source spans or regions · reject unsupported entities, ambiguous references, and questions answerable from OCR text without the relevant visual structure when image dependence is required
Maps, floorplans, and spatial layouts	Coordinates, routes, rooms, grids, compass marks, scale bars, landmarks	Shortest/longest path · direction and relative-position query · scale conversion · region inclusion/exclusion · distance aggregation · coordinate transform · route constraint satisfaction	Require visible scale, grid, or landmark evidence · reject unverifiable navigation/world-knowledge facts and missing-start/end-point prompts
Science process diagrams	Arrows, states, labels, apparatus, biological/chemical/physical processes	Symbol-component binding · process direction inference · conservation or constraint application · state-change comparison · controlled-variable reasoning · cause-effect chain extraction · diagram-plus-equation coupling	Require figure-specific evidence · reject pure domain trivia and mechanisms not represented in the diagram
Circuits, mechanics, and systems diagrams	Components, wires, forces, pulleys, blocks, gears, flows, switches	Equivalent-structure reduction · series/parallel or force-balance reasoning · state toggling · conservation constraints · component removal/addition counterfactual · input-output relation derivation	Require component-level grounding · reject prompts whose answer depends on unstated physical parameters
Natural quantitative scenes	Countable objects, spatial relations, occlusion cues, sizes, distances	Conditional counting · grouping and selection · size/distance comparison · occlusion-aware count · relation-chain reasoning · scene-to-arithmetic conversion · visible-ratio estimate	Require visible objects and relations · reject subjective attributes, pure commonsense, and objects too small or occluded to verify

Route	Trigger evidence	Evolution operators	Verification / rejection contract
Visual patterns, grids, and matrices	Repeated shapes, cells, transformations, colors, symmetry, missing entries	Pattern completion · rotation/reflection/translation query · symmetry or invariant detection · combinatorial count · row/column rule induction · missing-cell reasoning · transformation sequence inference	Require observable rule support · reject puzzles with multiple valid completions unless additional constraints make the answer unique
Multi-panel and temporal images	Ordered panels, before/after states, multiple views, sequence labels	Cross-panel delta · temporal ordering · causal state transition · common/different attribute comparison · aggregation across views · counterfactual within observed state changes · consistency across viewpoints	Require panel identity and ordering · reject changes not visible between panels or questions needing external video context
Graphs, networks, and trees	Nodes, edges, weights, arrows, hierarchies, adjacency, paths	Path length · reachability · degree/count query · subgraph comparison · hierarchy traversal · flow or dependency reasoning · tree-depth or least-common-ancestor query	Require visible node/edge labels · reject missing edge weights, hidden adjacency, and graph-theoretic assumptions not stated visually
Venn, set, and probability diagrams	Regions, overlaps, labels, counts, partitions, tree probabilities	Inclusion-exclusion · conditional probability · set difference/intersection · missing-region solve · consistency with total · probability-tree path aggregation	Require visible counts/probabilities · reject prompts with underdetermined region values or unstated independence assumptions
3D objects and perspective geometry	Solids, faces, nets, projections, depth cues, rotations	Surface/volume computation · net-to-solid mapping · view transformation · hidden-face inference when specified · spatial relation comparison · cross-section reasoning	Require visible dimensions or stated scale · reject hidden measurements unless inferable from given constraints
Timelines and event sequences	Time marks, durations, ordered events, intervals, dependencies	Duration aggregation · overlap comparison · before/after constraint solving · rate over time · missing event-time inference · schedule conflict query	Require visible timestamps or intervals · reject external historical facts and underdetermined schedules
Plots with uncertainty or measurement noise	Error bars, confidence bands, approximate ticks, noisy observations	Bound-aware comparison · interval overlap · significant-difference query · robust rank under uncertainty · approximate numeric extraction with tolerance	Require tolerance or interval answer · reject exact-value questions when visual uncertainty dominates

C.1 Reasoning-Skill Operators

Table 10 Route-independent reasoning-skill operators.

Operator	Prompt transformation	Acceptance signal
Multi-hop composition	Combine two or more visible facts before asking for the final answer	Intermediate facts can be cited and checked independently
Cross-region aggregation	Require evidence from separated regions, panels, rows, cells, or diagram components	Evidence metadata contains multiple grounded regions
Constraint injection	Add a condition that filters visible candidates or changes the computation path	Added condition is visually or textually supported and not contradictory
Inverse problem construction	Ask for an input, parameter, or missing quantity that would produce a visible output	Solver can verify by substituting the answer forward
Comparative reasoning	Convert direct reading into greater-than, difference, ratio, rank, or top- k comparison	Candidate answer is uniquely ranked after normalization
Extrema search	Ask for maximum, minimum, earliest, latest, nearest, farthest, or most/least frequent item	Search domain is finite and visible

Operator	Prompt transformation	Acceptance signal
Unit and scale conversion	Require conversion across axis scales, map scales, table units, percentages, or dimensional units	Unit mapping is stated or visible · normalized value passes tolerance
Rate and change reasoning	Ask for slope, percent change, temporal delta, growth, decay, or before/after comparison	Two or more grounded measurements support the change
Interpolation within support	Ask for an approximate value inside a visible interval or plotted range	Tolerance is specified · extrapolation is rejected unless explicitly bounded
Symbolic-numeric translation	Convert visual relations into equations, expressions, inequalities, or substitutions	Variables bind to visual/textual evidence and simplification agrees with the final answer
Combinatorial counting	Ask for counts under conditions, arrangements, paths, subsets, or grouped categories	Count domain is visually finite and all exclusions are explicit
Counterfactual within image constraints	Modify one visible condition and ask for the resulting answer while holding other visible conditions fixed	Counterfactual only changes stated variables · no external causal assumptions required
Distractor generation	Add options or false leads that require visual elimination rather than language priors	Each distractor has a known violated constraint
Evidence localization	Ask for the supporting cell, span, region, point, object, or diagram component in addition to the answer	Region/cell/span metadata is present and checkable
Difficulty calibration	Increase steps, add conditions, or compose operations only until the expected solver pass rate lies in the target range	Rollout pass rate remains between selected thresholds
Verifiability conversion	Prefer exact, normalized, bounded, or programmatically checkable answer forms over vague open-ended forms	A deterministic or high-confidence verifier can be attached
Ambiguity reduction	Rewrite open questions into constrained, unique-answer prompts with explicit scope and answer format	Multiple plausible interpretations are eliminated by schema and visual grounding
Adversarial visual dependence	Add a text-only distractor or require a visible detail that language priors alone cannot determine	Text-only probe fails or has low confidence

C.2 Answer-Type and Reward Contracts

Table 11 Answer-type-conditioned operators and reward contracts.

Answer route	Answer-conditioned evolution operators	Verification / reward contract
Multiple choice	Preserve legal options, regenerate more discriminative distractors, or convert direct lookup into option-wise constraint satisfaction	Final answer must be a valid option · verifier performs blind elimination over all options
Integer or decimal	Force arithmetic over visible quantities · add unit conversion · require rounded, exact, or bounded numeric form	Normalize units, signs, separators, and tolerance · programmatic check when possible
Fraction, ratio, percentage	Ask for normalized ratio, proportion, relative frequency, percentage point difference, or percent change	Canonicalize equivalent forms · verify numerator, denominator, and before/after quantities
Symbolic expression	Ask for expression derivation, simplification, parameter relation, or symbolic substitution into visual constraints	Run symbolic simplification or numeric substitution · bind each variable to evidence
Equation or inequality	Ask for an equation, threshold, feasible interval, or inequality relation satisfying visible constraints	Check algebraically or by sampled substitution · reject missing domain constraints
Coordinate, point, or interval	Ask for an intercept, coordinate, range, interval, region boundary, or approximate plotted value	Normalize coordinate frame · apply resolution-aware tolerance · require axis evidence

Answer route	Answer-conditioned evolution operators	Verification / reward contract
Set, list, or ranking	Ask for all objects satisfying a condition, ordered ranks, top- k , or grouped categories	Specify whether order matters · normalize aliases · verify completeness and unsupported-item absence
Boolean or verdict	Ask whether a visual-mathematical statement is true, false, satisfiable, increasing, connected, or consistent	Require a cited counterexample or supporting constraint · reject subjective yes/no prompts
Region, span, cell, or bounding evidence	Ask for the relevant crop, text span, table cell, plotted point, graph node, or diagram component supporting the answer	Require coordinates, cell IDs, OCR spans, graph IDs, or crop references in metadata
Short categorical answer	Convert open wording into a closed ontology such as color, shape, trend, state, relation type, component class, or direction	Normalize aliases · reject subjective labels or categories not visually grounded
Free-form rationale	Keep final answer short but require a concise evidence trace or step sketch	Use for SFT unless an extracted final answer has a deterministic checker
Program assertion	Ask for an answer that can be represented as executable assertions over extracted quantities	Reward is assertion pass/fail with format-error penalty

C.3 From Taxonomy to the 12-Topic Implementation

The design-space taxonomy above admits a wide combinatorial cross product of (visual route) \times (reasoning skill) \times (answer type) \times (verifier contract). Rather than instantiating this cross product at runtime, our current implementation discretizes the routing axes into 12 topic tags and pre-bundles, for each tag, the recommended reasoning operator, the recommended answer-type operator, and the matching verifier contract into a single evolution prompt template. Concretely, the 12 tags are: OCR, Image Description, Detection, Analysis, Content Creation, Suggestions, Summarization, Logical Reasoning, Science-Related, Concept Extraction, Medical Imaging, and Scene Understanding; their natural-language descriptions and example image families are documented in the released code. We chose this discretization for three reasons. First, a 12-topic vocabulary is small enough that each topic’s evolution prompt can be authored and audited by hand, and each gate’s failure modes can be enumerated; the full cross product would be impractical to maintain at this scale. Second, fixing the operator bundle per topic eliminates a class of compositional errors that arise when independently sampled operators produce internally inconsistent prompts (e.g., a multiple-choice answer-type operator attached to a problem whose answer is a continuous quantity). Third, adopting one prompt per topic makes the rejection reasons of the question-level gate easier to attribute back to the topic, which in turn supports per-topic threshold tuning. The trade-off is reduced expressivity: prompts that legitimately need to mix two routes (e.g., a document page combining a formula with a chart) are currently authored as part of the most fitting single topic rather than composed from independent route operators. Closing this gap with a verifier-aware operator-composition mechanism is left to future work.

C.4 HTV-Agent Decider Prompt (verbatim)

The conflict-aware decider in Section 3.3 is realized as a constrained LLM call (no tool use, no code execution). The system prompt below is the verbatim template used during VERIEVOL-RL construction; the user prompt is filled in with the question, optional context, options (if any), the solver’s hypothesized answer, and the verifier’s report, in that order.

System prompt (verbatim).

```
You are a final judge. You will see an initial answer from a Solver and a verification report from a
  Verifier. Your job is to decide the final answer.
Do not call tools, functions, notebooks, plugins, Python, or code interpreters.
Instructions:
1. Briefly summarize the verification’s conclusion (1--2 sentences).
```

2. Assess the QUALITY of the verification: is it logically sound, or does it contain self-contradictions, arithmetic errors, or unsupported claims?
3. Decide the final answer:
 - If verification is high-quality AND explicitly rejects the initial answer, trust the verification.
 - If verification is low-quality or self-contradictory, trust the initial answer.
 - If they agree, keep the answer.
4. If the Solver and Verifier disagree AND you are not confident in either, output `<require_rethink>true</require_rethink>`.
5. Output your exact final answer inside the tag `<final_answer>X</final_answer>`.
Examples: `<final_answer>A</final_answer>` or `<final_answer>42</final_answer>`.
6. Output your confidence as `<confidence>0--100</confidence>`.

User prompt template.

```

Question:
{question}
Context: {context} (omitted if empty)
Options: {choices} (omitted if not multiple choice)
Solver's answer: {hypothesis}
Verification report:
{verification_text}

```

Post-processing. The decider’s response is parsed by regular expressions for `<final_answer>`, `<confidence>`, and `<require_rethink>`. If `<require_rethink>true</require_rethink>` is present, the sample is marked as unresolved and rejected by the deterministic gate rather than being forced into either VERIEVOL-SFT or VERIEVOL-RL. The decider is queried with sampling temperature 0, so the decision is deterministic given the same solver/verifier inputs.

D Visual-Dependence Case Studies

Two illustrative flips. We describe two side-by-side reasoning-trace case studies referenced in Section 4.4. In the first, a geometry-diagram problem, RL-Origin asserts an unsupported angle while RL-Evol + Verifier extracts the correct measurement directly from the figure. In the second, a chart problem, RL-Origin reads the wrong column label while RL-Evol + Verifier identifies the column via its position relative to a labelled axis. Both pairs are drawn from the 12 paired flips identified in the 60-sample pilot (Layer 2, Section 4.4).

E Prompt-Evolution Case Studies

To make the effect of type-aware prompt evolution (§3.2) concrete, Figure 5 shows three seed–evolved pairs drawn from science-diagram seeds. In each case the **before** prompt is a single-step recognition or lookup query whose answer can often be retrieved from text priors or a single label, whereas the **after** prompt produced by the route-specific operators requires multi-step reasoning that is grounded in spatial relations, labels, and quantities that must be read from the image. The evolved answer is short and deterministically checkable, so it admits the reward-checker contract required for VERIEVOL-RL (§3.4). We annotate each evolved prompt with the key reasoning steps from the HTV-AGENT-verified solution.

F Iterative Construction Algorithm

Algorithm 1 specifies the iterative construction procedure used by VERIEVOL. It operates one seed sample at a time: after routing, candidate questions are generated under route-specific operators, filtered through the question-level gate ϕ_q , screened by HTV-AGENT, filtered again through the answer-level gate g , materialized into VERIEVOL-SFT, and (if the verifier supports deterministic checking and the rollout pass rate falls in the useful range) additionally materialized into VERIEVOL-RL.

Algorithm 1 Iterative construction of the SFT and RL training corpora.

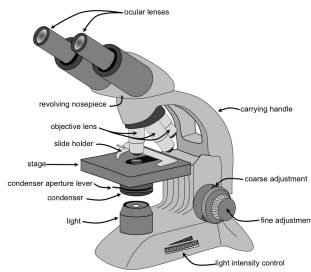
Require: seed pool \mathcal{D}_0 , evolution operator library \mathcal{O} , question gate ϕ_q , answer gate g , rollout budget N

Ensure: $\mathcal{D}_{\text{SFT}}, \mathcal{D}_{\text{RL}}$

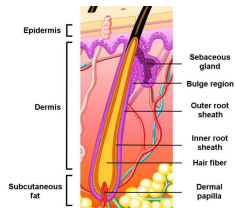
```
1: for each seed sample  $(I, q, a) \in \mathcal{D}_0$  do
2:   infer routing tags  $z = (t, y, u, c)$  using image, OCR, layout, and question features
3:   generate candidates  $\mathcal{Q} = \{q'_j\}_{j=1}^K$  with operators in  $\mathcal{O}_z$ 
4:   for each  $q'_j \in \mathcal{Q}$  do
5:     if  $\phi_q(I, q, q'_j) = 0$  then
6:       discard or regenerate  $q'_j$ ; continue
7:     end if
8:     run HTV-AGENT to obtain verified answer  $\hat{a}_j$ , accepted reasoning trace  $r_j$ , evidence  $e_j$ , verifier  $v_j$ ,
metadata  $m_j$ 
9:     if  $g(\hat{a}_j, m_j) = 0$  then
10:      discard  $q'_j$ ; continue
11:     end if
12:     materialize  $(I, q'_j, r_j, \hat{a}_j, e_j)$  into VERIEVOL-SFT
13:     if  $v_j$  supports deterministic or high-confidence checking then
14:       estimate rollout pass rate  $\rho_j$  with  $N$  student samples
15:       if  $\rho_{\min} < \rho_j < \rho_{\max}$  then
16:         add  $(I, q'_j, \hat{a}_j, v_j, m_j)$  to VERIEVOL-RL
17:       end if
18:     end if
19:   end for
20: end for
```



Lunar-phase diagram



Labeled compound microscope



Skin / hair-follicle anatomy

Before evolution – seed prompt

Q: Identify the moon phase in which the whole side facing earth is lit. A: Full Moon

After evolution – image-grounded reasoning prompt

Q: Assuming the synodic period of the Moon is exactly 29.5 days, calculate the total duration, in days, that the visible illumination is both strictly greater than the percentage shown during the **First Quarter Half** phase and strictly less than the percentage shown during the **Full Moon** phase.

Reasoning: First Quarter = 50%, Full Moon = 100%; the band $50% < P < 100%$ spans exactly the Waxing- and Waning-Gibbous quarters, i.e. $T/2 = 29.5/2$.

A: 14.75

Before evolution – seed prompt

Q: Where would you look through this microscope? A: Ocular lenses

After evolution – image-grounded reasoning prompt

Q: Starting from a microscope that is powered off, unfocused, and set to the lowest-power objective, what is the minimum number of **distinct, labeled mechanical controls** (knobs, levers, sliders) that must be manipulated to obtain a focused, properly illuminated view at the second-highest magnification?

Reasoning: distinct controls used = {light-intensity control, coarse adjustment, fine adjustment, condenser-aperture lever, revolving nosepiece}; re-uses do not add new controls.

A: 5

Before evolution – seed prompt

Q: In which layer of the skin does the sebaceous gland lie? A: Dermis

After evolution – image-grounded reasoning prompt

Q: Compute the ratio of (A) the number of labeled hair-follicle components that physically extend past the Dermis/Subcutaneous-fat boundary, to (B) the number of labeled structures whose entirety is confined strictly within the Dermis.

Reasoning: $A = 4$ (outer/inner root sheath, hair fiber, dermal papilla); $B = 2$ (sebaceous gland, bulge region); ratio = $4/2$.

A: 2

Figure 5 Type-aware prompt evolution on science-diagram seeds. For each case, the grey box is the original single-step recognition seed and the purple box is the evolved, image-grounded multi-step reasoning prompt produced by the route-specific operators (§3.2), annotated with the key steps of its HTV-AGENT-verified solution. Evolved answers remain short and deterministically checkable, satisfying the reward-checker contract for VERIEVOL-RL.