LEARNING FROM SUCCESSES AND FAILURES: EFFICIENT SELF-EVOLUTION THROUGH ASYMMETRIC SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning models can continue to improve after instruction tuning, yet existing approaches often either use reinforcement learning for self-evolution or rely on pre-existing per-question data for supervised refinement. We ask whether an instruction-tuned model can improve by directly learning from the structure of its own generated experience. Our key insight is asymmetric supervision: verified successes provide constructive targets for SFT, while competing incorrect answers provide comparative signals without treating failed reasoning trajectories as negative demonstrations. Starting only from a high-level domain prior and difficulty specifications, our framework constructs capability-matched curricula from model-generated problems and converts verified experience into success amplification and answer-level failure discrimination, using parameter-efficient LoRA updates on a compact training set. On Qwen3-4B and Qwen3-8B, verified-success SFT alone improves average accuracy across five challenging mathematical benchmarks by 2.27 and 2.28 percentage points, demonstrating that self-improvement without reinforcement learning is possible. Failure-aware supervision provides further gains, yielding total improvements of 3.12 and 2.97 percentage points, respectively, with aggregate gains across a broader nine-benchmark mathematical reasoning suite. Repeated-sampling analyses further indicate improved access to successful solutions.

1 INTRODUCTION

Can reasoning models autonomously construct and learn from their own training experience? Recent self-evolving reasoning methods suggest that they can. Absolute Zero (Zhao et al., 2026a), Self-Questioning Language Models (Chen et al., 2025), and R-Zero (Huang et al., 2026) allow language models to generate or adapt training problems and improve through reinforcement-learning mechanisms driven by self-generated rewards. These approaches reduce dependence on fixed human-curated task corpora by making problem generation part of learning. Their capability updates, however, primarily convert self-generated experience into rewards for policy optimization, motivating direct learning from the richer structure of generated and verified reasoning trajectories.

Model-generated reasoning also provides effective supervision. STaR (Zelikman et al., 2022), Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023), and ReST^{EM} (Singh et al., 2023) reuse successful generations through supervised fine-tuning, while Self-Verified Distillation (SVD) (Lee & Liang, 2026) enables a post-trained model to generate, verify, and learn from its own solutions over unlabeled seed questions. These methods establish successful trajectories as useful targets, but typically assume an externally supplied question distribution and center supervision on successful solutions. Autonomous self-evolution instead produces both the training problems and diverse reasoning outcomes over them. We ask whether an instruction-tuned reasoner can generate capability-matched problems, verify its own experience, and improve by exploiting the different information carried by successful and failed generations.

Our central observation is that these two forms of experience provide asymmetric but complementary supervision. A verified success is a complete constructive demonstration containing a valid reasoning process and correct outcome, making it suitable for SFT. A failed trajectory may still contain

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

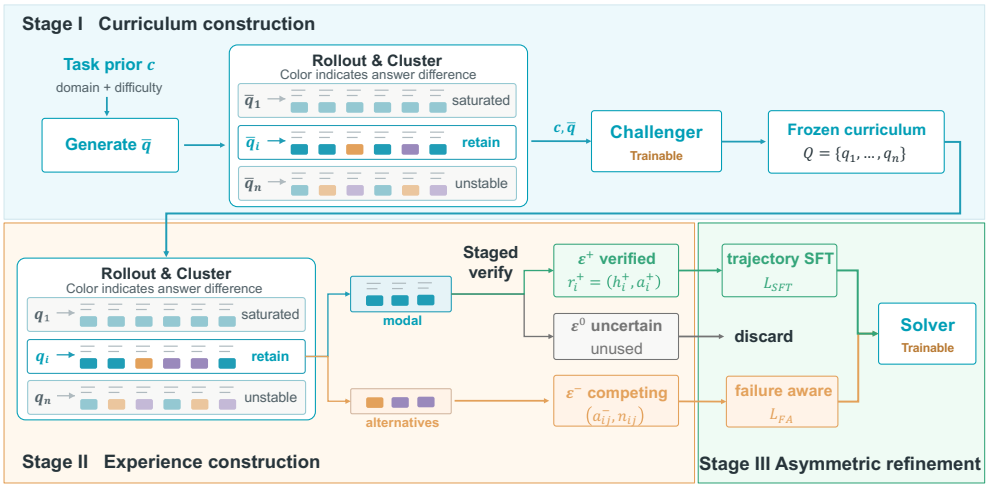


Figure 1: **Overview of our self-evolution framework with asymmetric supervision. Stage I** bootstraps a Challenger from initial model M_0 on capability-matched self-generated questions and freezes its curriculum $\mathcal{Q}$. **Stage II** uses the unchanged M_0 to construct verified successes $\mathcal{E}^+$, recurrent competing alternatives $\mathcal{E}^-$, and discarded uncertain experience $\mathcal{E}^0$ through repeated solving, answer clustering, and staged verification. **Stage III** trains a fresh Solver from M_0 asymmetrically: verified trajectories provide trajectory-level SFT, while failure-aware learning contrasts verified answers with recurrent alternatives under the same verified reasoning context.

correct problem interpretation and useful intermediate reasoning before a localized error, so treating it as a complete negative demonstration introduces ambiguous credit assignment. Recurrent failures instead identify competing outcomes that should be disfavored relative to a verified answer. We therefore use verified successes as trajectory-level constructive targets, recurrent failures as answer-level comparative signals, and exclude uncertain experience from optimization. Self-verification thus determines both which experience is retained and how it is learned from.

We instantiate this principle through curriculum construction, experience verification, and asymmetric Solver refinement (Figure 1). The initial model generates candidate mathematical problems from high-level domain and difficulty specifications and estimates their difficulty through repeated solving. Capability-matched problems are used to specialize a Challenger that constructs the final curriculum, which is then frozen. The unchanged original model repeatedly solves each problem, answers are grouped by mathematical equivalence, and staged self-verification filters candidate successful trajectories. A fresh Solver initialized from the original model uses verified trajectories for SFT, amplifying successful reasoning modes already accessible to the model. A failure-aware pairwise objective additionally contrasts verified answers with recurrent alternatives under the same verified reasoning context, combining constructive learning from success with comparative learning from failure.

Experiments on Qwen3-4B and Qwen3-8B consistently support this design. Verified-success SFT alone improves both Math-9 and Hard-5 over the initial models, showing that self-generated constructive supervision is sufficient for measurable self-improvement without a reinforcement-learning Solver update. Incorporating recurrent failure signals from the same collected experience improves both aggregates further. The full framework improves Math-9 by 2.29 and 2.11 percentage points and Hard-5 by 3.12 and 2.97 points on Qwen3-4B and Qwen3-8B, respectively. These gains support complementary information from recurrent failures beyond positive imitation.

We make three main contributions.

- **Self-evolution from verified successes without reinforcement learning.** We show that verified-success SFT alone suffices to improve an instruction-tuned reasoner using autonomously generated problems and self-verified solutions, without a pre-existing per-question training corpus or a reinforcement-learning Solver update.

- **Asymmetric supervision from successes and failures.** We operationalize the distinct information carried by self-generated successes and failures. Verified successes provide trajectory-level constructive targets, whereas recurrent failures provide answer-level comparative supervision through a failure-aware pairwise objective under the same verified reasoning context, avoiding direct negative training on complete failed trajectories.

- **Effective utilization of self-generated experience.** Across two model scales, verified-success SFT produces consistent gains and recurrent failure supervision improves performance further, supporting efficient self-evolution when different forms of experience are verified and converted into supervision according to the information they reliably provide.

2 Related Work

2.1 Self-Training, Synthetic Supervision, and Verification

Self-training methods such as STaR (Zelikman et al., 2022), RFT (Yuan et al., 2023), ReST^{EM} (Singh et al., 2023), and related self-improvement approaches reuse model-generated correct or high-confidence reasoning traces as training targets (Huang et al., 2023). Self-Instruct (Wang et al., 2023) and MetaMath (Yu et al., 2024) expand supervision through generated instructions or transformed questions, while SPIN (Chen et al., 2024) and Self-Rewarding Language Models (Yuan et al., 2024) derive iterative signals from model-generated responses or judgments. More recent self-distillation methods such as SVD (Lee & Liang, 2026) and SD-Zero (He et al., 2026) further convert model-generated solutions or revisions into training supervision. Verification work complements this direction through outcome and process verifiers (Cobbe et al., 2021; Lightman et al., 2023; Wang et al., 2024a). These studies show that self-generated experience can support learning when reliable supervision can be extracted, but typically operate on externally seeded task distributions or do not couple autonomous curriculum construction with the learning mechanism.

2.2 Autonomous Self-Evolution and Curriculum Construction

Absolute Zero (Zhao et al., 2026a), Self-Questioning Language Models (Chen et al., 2025), and R-Zero (Huang et al., 2026) move task generation inside the learning loop, using self-generated verifiable or capability-matched problems to drive reward-based self-evolution. Subsequent work studies human grounding, curriculum decoupling, diversity, and diagnosis-guided task generation (Yu et al., 2025; Fan et al., 2026; Li et al., 2026; Wei et al., 2026). Related methods reduce external supervision on fixed unlabeled questions through majority-based pseudo-rewards, self-consistency, confidence, or entropy-derived signals (Zuo et al., 2026; Shafayat et al., 2025; Zhao et al., 2026b; Agarwal et al., 2026). Our setting is closest to autonomous curriculum methods, but differs in the learning interface: rather than primarily converting self-generated experience into rewards for RL, we verify its structure and directly map different experience types to different forms of supervision.

2.3 Comparative Learning and Learning from Failures

DPO (Rafailov et al., 2023) and SimPO (Meng et al., 2024) establish pairwise preference optimization, while ScPO (Prasad et al., 2024) derives preferences from self-consistency on unlabeled reasoning problems. Recent work also shows that unsuccessful trajectories can remain informative: negative reasoning samples may retain useful intermediate structure, failed attempts can be converted into correction supervision, and negative-sample reinforcement can improve reasoning (Xueyun et al., 2026; Ren et al., 2026; Zhu et al., 2026). These findings motivate using failure information without treating an entire failed trajectory as uniformly undesirable. Our method combines this insight with autonomous experience generation: verified successes provide constructive trajectory-level targets learned by SFT, whereas recurrent competing failures provide answer-level comparative supervision. The pairwise form is related to reference-free preference objectives such as SimPO; our contribution lies in the asymmetric conversion of self-generated, self-verified experience rather than a new generic preference loss.

3 METHODOLOGY

3.1 OVERVIEW

Our framework follows a simple principle: the model generates its own problems, solves them, and learns differently from successful and failed attempts. Verified successes are learned through SFT, while recurrent incorrect answers provide complementary supervision through an answer-level pairwise objective.

Specifically, we study autonomous post-instruction self-evolution from an instruction-tuned reasoning model M_0, using only a high-level task prior specifying the target domain and difficulty. The framework has three stages. First, M_0 bootstraps a capability-aware Challenger, which generates a curriculum matched to the model's current reasoning boundary. The curriculum is frozen and the Challenger adapter is discarded. Second, the unchanged M_0 repeatedly solves each curriculum question; answer clustering and staged verification convert these rollouts into verified successes $\mathcal{E}^+$, recurrent competing failures $\mathcal{E}^-$, and uncertain experience $\mathcal{E}^0$. Third, a fresh Solver initialized from M_0 learns asymmetrically from retained experience: verified successes provide trajectory-level SFT targets, whereas recurrent competing answers provide answer-level comparative supervision. Uncertain experience is excluded from optimization. Thus, the Challenger determines which problems are practiced, while Solver refinement is grounded only in experience generated by the original model.

3.2 AUTONOMOUS CURRICULUM CONSTRUCTION

Capability-aware bootstrap. Let c denote a condition sampled from the task prior, and let $\bar{q}$ denote a bootstrap candidate question generated by M_0 conditioned on c. We distinguish the bootstrap budget used to train the Challenger from the later Solver collection. For each candidate $\bar{q}$, the original model produces K_{C} independent trajectories, whose final answers are merged into m equivalence clusters with counts $c_1, \ldots, c_m$, where $\sum_{j=1}^{m} c_j = K_{\mathrm{C}}$. We define Challenger-stage answer consistency as

$$\rho_{\mathrm{C}}(\bar{q}) = \frac{\max_{1 \le j \le m} c_j}{K_{\mathrm{C}}}. \tag{1}$$

A question is treated as capability-matched only if it has a unique modal cluster and

$$\tau_{\min} < \rho_{\mathrm{C}}(\bar{q}) \le \tau_{\max}. \tag{2}$$

Lower consistency indicates unstable answers, whereas higher consistency indicates questions already close to saturation for M_0. After filtering, the Challenger is trained on N_{C} retained condition–question $(c, \bar{q})$ pairs.

Curriculum generation. The trained Challenger generates the frozen Solver curriculum $\mathcal{Q} = \{q_i\}_{i=1}^{N_{\mathrm{S}}}$. We then discard the Challenger adapter. All subsequent Solver trajectories and verification signals are produced by the original model M_0, so Challenger adaptation affects problem selection but not the model that supplies Solver experience.

3.3 SELF-GENERATED EXPERIENCE CONSTRUCTION

For each $q_i \in \mathcal{Q}$, the original M_0 produces K_{S} independent reasoning trajectories whose final answers are grouped into equivalence clusters. We first apply the same normalized-count consistency criterion used above to the unique modal cluster, retaining only questions whose modal frequency lies within the capability thresholds. Because answer agreement alone does not establish correctness, we then apply programmatic checks and staged self-verification to the modal trajectories (Appendix C). Let v_i denote the number of modal trajectories that pass all verification stages. We reapply the same consistency criterion after verification and retain the question only if $\tau_{\min} < v_i/K_{\mathrm{S}} \le \tau_{\max}$. The verified modal trajectories form $\mathcal{E}^+$, while modal candidates that fail verification are assigned to $\mathcal{E}^0$ and are neither imitated nor explicitly discouraged. Valid non-modal answer clusters form $\mathcal{E}^-$ and retain their occurrence counts as recurrent competing outcomes. Questions that fail either consistency check contribute no Solver training target.

For each question with a verified success, we select one complete successful trajectory $r_i^+ = (h_i^+, a_i^+)$, where h_i^+ is the verified reasoning prefix before the final answer and a_i^+ is the verified answer. The full trajectory r_i^+ is used for SFT. For the same question, we retain J_i competing

clusters $\{(a_{ij}^-, n_{ij})\}_{j=1}^{J_i}$, where a_{ij}^- is the answer represented by the j-th cluster and n_{ij} is its rollout count. Importantly, failure-aware training uses these competing answers and their frequencies rather than treating complete failed reasoning trajectories as negative demonstrations.

An illustrative example. Consider the question: "How many four-digit integers have strictly increasing digits?" The correct answer is $\binom{9}{4} = 126$, since zero cannot appear and each choice of four digits from 1 to 9 has exactly one increasing arrangement.

Suppose eight Solver rollouts produce the answer 126 five times and three competing answers, 210, 84, and 120, once each. Four of the five modal trajectories pass staged verification, yielding pre- and post-verification consistency scores of $5/8$ and $4/8$, respectively, both within our capability-matching range.

We select one complete verified trajectory for SFT and use the three competing answers for failure-aware supervision, each with weight $1/3$. Under the same verified reasoning prefix, the failure-aware objective favors 126 over the competing answers without penalizing their original reasoning trajectories. The unverified modal trajectory is discarded.

3.4 Asymmetric Solver Refinement

Learning from verified successes. Let $\mathcal{I}^+$ be the indices of curriculum questions with a verified successful trajectory. We optimize

$$\mathcal{L}_{\mathrm{SFT}} = -\frac{1}{|\mathcal{I}^+|} \sum_{i \in \mathcal{I}^+} \log p_\theta(r_i^+ \mid q_i), \tag{3}$$

where θ denotes the Solver parameters. Because each r_i^+ was already reachable from M_0, this objective assigns higher probability to verified successful modes that the initial model may sample unreliably; we refer to this effect as *success amplification*.

Learning from recurrent failures. A failed trajectory can contain useful intermediate reasoning, so its incorrect final answer does not justify suppressing the entire sequence. We therefore use only the reliable comparative signal: under the same verified positive reasoning prefix, the verified answer should outrank recurrent competing answers. For an answer $a = (a_1, \ldots, a_{|a|})$, define the length-normalized log-probability

$$s_\theta(a \mid q_i, h_i^+) = \frac{1}{|a|} \sum_{t=1}^{|a|} \log p_\theta(a_t \mid q_i, h_i^+, a_{<t}), \tag{4}$$

where $|a|$ is the answer length and $a_{<t}$ denotes its prefix before token t. Let $\mathcal{I}_{\mathrm{FA}} = \{i \in \mathcal{I}^+ : J_i > 0\}$ index the questions with at least one competing answer. The failure-aware objective is

$$\mathcal{L}_{\mathrm{FA}} = -\frac{1}{|\mathcal{I}_{\mathrm{FA}}|} \sum_{i \in \mathcal{I}_{\mathrm{FA}}} \sum_{j=1}^{J_i} \omega_{ij} \log \sigma\big(\beta \left[s_\theta(a_i^+ \mid q_i, h_i^+) - s_\theta(a_{ij}^- \mid q_i, h_i^+)\right]\big), \tag{5}$$

where σ is the sigmoid function, $\beta > 0$ scales the pairwise score gap, and $\omega_{ij} = n_{ij} / \sum_{\ell=1}^{J_i} n_{i\ell}$ weights each competing answer by its recurrence among retained failure clusters. Thus, frequent competing outcomes receive more comparative weight without assigning negative credit to their full reasoning traces.

Combined objective. The final Solver objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{SFT}} + \lambda \mathcal{L}_{\mathrm{FA}}, \tag{6}$$

where $\lambda \geq 0$ controls the contribution of failure-aware supervision. The two terms use self-generated experience at different granularities: SFT amplifies complete verified successes, while the failure-aware term discriminates verified answers from recurrent competing outcomes. This constitutes the asymmetric supervision used for Solver refinement.

4 EXPERIMENTAL SETUP

4.1 MODELS AND TRAINING

We evaluate Qwen3-4B and Qwen3-8B in thinking mode (Yang et al., 2025). For each scale, *Initial* is the original post-trained checkpoint, *Success-SFT* trains the Solver using only verified successes $\mathcal{E}^+$, and *Full* additionally incorporates recurrent failures $\mathcal{E}^-$ through $\mathcal{L}_{\mathrm{FA}}$. Success-SFT and Full use the same Solver curriculum and collected experience, and all variants share the same evaluation protocol. We set $K_{\mathrm{C}} = K_{\mathrm{S}} = 8$ and $\tau_{\min} = 0.25$, $\tau_{\max} = 0.75$, so a unique modal answer must occur in 3–6 rollouts; during Solver collection, 3–6 modal trajectories must also remain verified after staged verification. For both model scales, $N_{\mathrm{C}} = N_{\mathrm{S}} = 2000$. Challenger and Solver are adapted with rank-16 LoRA for one epoch; remaining optimization, decoding, verification, and implementation settings are provided in the appendix.

4.2 BENCHMARKS AND METRICS

We evaluate mathematical reasoning on AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), AIME26 (Zhang & Math-AI, 2026), HMMT25 (Dekoninck et al., 2026), AMC23, MATH500 (Lightman et al., 2023), Minerva Math (Lewkowycz et al., 2022), GSM8K (Cobbe et al., 2021), and OlympiadBench(English-only math subset) (He et al., 2024). *Math-9* is the unweighted average over all nine benchmarks, while *Hard-5* averages AIME24, AIME25, AIME26, HMMT25, and OlympiadBench. Hard-5 emphasizes competition- and olympiad-style problems requiring longer and less routine reasoning, matching the high-difficulty regime targeted by our self-generated curriculum, and is used consistently for difficult-regime accuracy and rollout analysis. Additionally, we evaluate GPQA-Diamond (Rein et al., 2023) for cross-domain scientific reasoning transfer and MMLU-Pro (Wang et al., 2024b) for broader capability preservation.

All reported results are obtained over multiple evaluation runs. Accuracy is the primary metric for main results and ablations. On Hard-5, Pass@k measures the probability of obtaining at least one correct solution within k samples and therefore captures *successful-trajectory accessibility*. We further classify repeated-sampling outcomes as *Unsolved* when no trajectory is correct, *Mixed-success* when correct and incorrect trajectories coexist, and *Mastered* when all trajectories are correct, separating solution reachability from consistency.

5 RESULTS AND ANALYSIS

We evaluate whether verified successes alone enable self-evolution and recurrent failures add complementary value, followed by analyses of repeated sampling, supervision and experience-construction ablations, and generalization beyond mathematics.

5.1 VERIFIED SUCCESSES ENABLE SELF-EVOLUTION

Table 1 compares Initial, Success-SFT, and Full to test whether verified-success SFT enables self-evolution and whether failure-aware supervision provides additional gains.

Success-SFT improves both Hard-5 and Math-9 at both model scales, gaining 2.27 and 2.28 percentage points on Hard-5 for Qwen3-4B and Qwen3-8B. Full attains the strongest aggregates, with Initial-to-Full gains of 3.12/2.29 points on Hard-5/Math-9 for 4B and 2.97/2.11 for 8B. Thus, verified successes alone suffice for measurable self-evolution without a reinforcement-learning Solver update, while recurrent failures improve aggregate performance further. Individual benchmarks are not uniformly monotonic, so the evidence supports complementary supervision across tasks rather than universal per-benchmark dominance.

5.2 COMPLEMENTARY VALUE OF RECURRENT FAILURES

Full improves over Success-SFT on both aggregates at both scales while using the same Solver curriculum, collected experience, and verified-success targets. The additional gains therefore support the value of recurrent failures beyond positive imitation. The supervision ablations in Table 4 reinforce this interpretation, with the combined objective outperforming either verified-success SFT or

Table 1: Mathematical reasoning accuracy for Qwen3-4B and Qwen3-8B before refinement (Initial), after training only on verified successes (Success-SFT), and after the complete asymmetric-supervision objective (Full). Hard-5 averages AIME24, AIME25, AIME26, HMMT25, and OlympiadBench, while Math-9 averages all nine mathematical benchmarks. All values are percentages. **Bold** marks the best result per column within each model scale.

Method	AIME24	AIME25	AIME26	HMMT25	AMC23	MATH500	Minerva	GSM8K	Olympiad	Hard-5	Math-9
Qwen3-4B											
Initial	72.33	60.33	62.00	26.33	95.25	95.60	26.84	**95.07**	68.40	57.88	66.91
Success-SFT	73.00	62.67	66.33	29.00	96.00	**97.00**	26.10	95.00	**69.73**	60.15	68.31
Full	**75.00**	**63.67**	**66.67**	**30.67**	**96.50**	96.80	**29.41**	**95.07**	68.99	**61.00**	**69.20**
Qwen3-8B											
Initial	73.33	66.67	62.00	28.00	95.50	96.00	29.04	95.75	69.88	59.98	68.46
Success-SFT	**77.33**	67.00	66.67	29.67	94.75	96.80	29.41	95.45	70.62	62.26	69.74
Full	76.00	**68.00**	**67.67**	**32.33**	**96.25**	**97.40**	**30.88**	**95.83**	**70.77**	**62.95**	**70.57**

Table 2: Successful-trajectory accessibility on Hard-5 as the rollout budget increases. Pass@k estimates the probability that at least one of k sampled trajectories solves a problem correctly. Comparing Initial, Success-SFT, and Full distinguishes the behavioral changes associated with verified-success learning and the complete asymmetric-supervision objective. All values are percentages. **Bold** marks the best result at each rollout budget within a model scale.

Method	Pass@1	Pass@2	Pass@4	Pass@8	Pass@10
Qwen3-4B					
Initial	57.88	64.74	68.34	70.85	71.43
Success-SFT	60.15	66.84	**71.27**	**74.07**	**74.73**
Full	**61.00**	**66.90**	70.68	73.43	74.12
Qwen3-8B					
Initial	59.98	67.54	71.45	73.62	74.24
Success-SFT	62.26	68.58	**72.90**	**76.06**	**77.00**
Full	**62.95**	**69.11**	72.80	75.41	76.33

failure-aware training alone. Recurrent failures thus complement, rather than replace, the constructive signal supplied by verified successes.

5.3 Successful-Trajectory Accessibility

Single-response accuracy cannot distinguish an absent successful mode from one that is merely sampled unreliably. We therefore analyze Hard-5 with Pass@k and problem-level rollout states.

Accessibility across rollout budgets. Table 2 shows that Success-SFT exceeds Initial at every measured rollout budget for both scales, indicating improved access to successful trajectories under repeated sampling. Full is strongest at Pass@1 and Pass@2, whereas Success-SFT is slightly higher at larger k. Thus, both refinements improve accessibility over Initial, with the full objective most favorable at low sampling budgets and Success-SFT retaining greater high-budget reachability.

Problem-level rollout states. Table 3 complements Pass@k by separating reachability from consistency. Success-SFT has the lowest Unsolved rate at both scales, while Full has the largest Mastered region and reduces Mixed-success relative to Success-SFT. Together, these results suggest that verified-success learning broadens the set of problems with reachable correct solutions, whereas the complete objective is associated with more consistently correct outcomes under repeated sampling. We treat this as a behavioral interpretation, since these diagnostics do not identify how probability mass moves among individual reasoning modes.

5.4 Ablation Studies

We test two sources of selectivity, separating how experience is constructed before training from how retained experience is converted into supervision. All main-text ablations use Qwen3-4B and report Hard-5 and Math-9; Appendix E provides the complete per-benchmark results.

Table 3: Problem-state distribution on Hard-5 under repeated sampling. Unsolved denotes problems for which no trajectory is correct, Mixed-success denotes problems with both correct and incorrect trajectories, and Mastered denotes problems for which every trajectory is correct. The state distribution complements Pass@k by separating reachability from consistency. Values are percentages. **Bold** marks the lowest Unsolved rate and highest Mastered rate within each model scale.

Method	Unsolved	Mixed-success	Mastered
Qwen3-4B			
Initial	28.57	35.61	35.82
Success-SFT	**25.27**	37.37	37.37
Full	25.87	34.18	**39.94**
Qwen3-8B			
Initial	25.75	33.89	40.36
Success-SFT	**23.00**	38.44	38.57
Full	23.67	35.34	**41.00**

Table 4: Supervision-design ablations on Qwen3-4B. The variants compare positive-target selection, the contribution of constructive success supervision, and answer-level versus trajectory-level use of recurrent failures. Full denotes the complete asymmetric-supervision objective. All values are percentages on the unified Hard-5 and Math-9 aggregates.

Method	Hard-5	Math-9
Reference		
Full	**61.00**	**69.20**
Positive-target controls		
Success-SFT	60.15	68.31
All-positive SFT	58.14	67.33
Failure-use controls		
Only-FA	58.89	67.61
Trajectory-level FA	59.19	67.52

Supervision design. Table 4 compares positive-target multiplicity, the contribution of constructive success supervision, and the granularity of failure comparison. Success-SFT uses one selected verified trajectory per question, All-positive SFT uses all verified positives, Only-FA removes constructive SFT, Trajectory-level FA compares complete positive and failed trajectories, and Full combines the compact success target with answer-level failure-aware supervision.

Using all verified positives underperforms Success-SFT, showing that retaining more positive trajectories is not inherently better. Only-FA also trails Success-SFT, whereas Full outperforms both, supporting constructive success learning as the foundation and recurrent failures as complementary supervision. Trajectory-level FA remains below Full, providing supporting evidence for comparing answers under a verified reasoning context rather than assigning negative credit to complete failed reasoning traces.

Experience construction. Table 5 weakens one component of experience construction at a time: Majority-only removes staged verification, Coupled C./S. removes Challenger/Solver role isolation, No Challenger removes Challenger specialization, and Random selection removes capability-aware problem selection.

Every construction ablation lowers both aggregates, with Random selection producing the largest degradation. Weaker verification, Challenger/Solver coupling, and removing Challenger specialization also reduce performance. These results show that the value of self-generated experience depends on which problems and trajectories are retained and how reliably they are verified; role isolation is a useful design choice within this pipeline. Together with the supervision ablations, they support selective construction, verification, and asymmetric use of self-generated experience.

Table 5: Experience-construction ablations on Qwen3-4B, evaluating staged verification, Challenger/Solver role isolation, Challenger specialization, and capability-aware selection under the Full training objective. Hard-5 and Math-9 accuracies are reported as percentages.

Method	Hard-5	Math-9
Reference		
Full	**61.00**	**69.20**
Verification control		
Majority-only	59.64	68.22
Curriculum and role controls		
Coupled C./S.	58.87	67.53
No Challenger	59.98	68.41
Random selection	58.14	66.81

Table 6: Performance beyond the mathematical training domain. GPQA-Diamond probes scientific reasoning outside the mathematical training domain, while MMLU-Pro is used as a broader capability-preservation check. The comparison is between the original checkpoint and the complete Full model at each scale. All values are percentages. **Bold** marks the higher GPQA-Diamond result within each model scale.

Method	GPQA-Diamond	MMLU-Pro
Qwen3-4B		
Initial	57.07	61.72
Full	**62.63**	**62.12**
Qwen3-8B		
Initial	60.61	65.18
Full	**65.15**	**65.23**

5.5 Generalization Beyond Mathematical Reasoning

Because the self-generated curriculum is entirely mathematical, Table 6 compares Initial and Full on GPQA-Diamond for cross-domain scientific reasoning and MMLU-Pro for broader capability preservation.

Full records higher GPQA-Diamond accuracy at both scales. Because GPQA-Diamond is absent from the self-generated curriculum, this suggests that refinement may extend beyond mathematics. On MMLU-Pro, the performance remains comparable or slightly better, with no observed degradation on this broader capability check. These observations complement the mathematical reasoning results.

6 Conclusion

We study autonomous reasoning self-evolution by converting self-generated and self-verified experience into asymmetric supervision. Verified successes provide constructive demonstrations and can be learned through SFT, whereas recurrent competing outcomes provide complementary comparative signals without treating complete failed trajectories as negative demonstrations.

Experiments show that verified-success SFT consistently improves mathematical reasoning without a reinforcement-learning Solver update, while incorporating recurrent failure signals yields further gains in the complete framework. Repeated-sampling results show improved accessibility of successful solutions, and ablations indicate that gains depend on how synthetic experience is selected, verified, and converted into supervision under the tested construction and supervision settings. Overall, these findings support learning from self-generated experience according to the information it can reliably provide: verified successes as constructive supervision and recurrent failures as complementary comparative supervision.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic mathematical problems and reasoning trajectories as part of the proposed framework, implementing parts of the method and debugging experimental scripts, and assisting with the analysis and interpretation of experimental results.

We have not used generative AI tools for conceptual framing, hypothesis formulation, research methodology or experimental design, formulation or proof of mathematical claims, translation, or dataset cleaning and reformatting, and the rest of the required disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for searching for and summarizing related literature, assisting with manuscript drafting, improving readability and grammar, and organizing references.

We have reviewed all AI-assisted work. AI-assisted code and experimental scripts were tested, synthetic training data were filtered and verified using the procedures described in this paper, and experimental results, analyses, references, and manuscript text were reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed implementation and training settings in Appendix A, including curriculum generation, Solver rollout collection, LoRA configurations, and optimization hyperparameters. Appendix B presents the prompt templates, while Appendix C describes the staged self-verification procedure and the construction of successful, failed, and uncertain experience. We also provide an illustrative training example in Appendix D. The main text specifies the evaluation benchmarks, metrics, and controlled ablations. We will release our code and experimental configurations upon acceptance.

ETHICS STATEMENT

This work is a methodological study of autonomous reasoning self-evolution using model-generated mathematical problems and publicly available evaluation benchmarks. It does not involve human-subject experiments or the collection of new personal data.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *Advances in Neural Information Processing Systems*, 38:107150–107180, 2026.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Shengda Fan, Xuyan Ye, and Yankai Lin. Darc: Decoupled asymmetric reasoning curriculum for llm evolution. *arXiv preprint arXiv:2601.13761*, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-distillation zero: Self-revision turns binary rewards into dense supervision. *arXiv preprint arXiv:2604.12002*, 2026.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. In *International Conference on Learning Representations*, volume 2026, pp. 130770–130790, 2026.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Tony Lee and Percy Liang. Self-verified distillation: Your language model is secretly its own synthetic data pipeline. *arXiv preprint arXiv:2605.26132*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Gengsheng Li, Jinghan He, Shijie Wang, Dan Zhang, Ruiqi Liu, Renrui Zhang, Zijun Yao, Junfeng Fang, Haiyun Guo, and Jinqiao Wang. R-diverse: Mitigating diversity illusion in self-play llm training. *arXiv preprint arXiv:2602.13103*, 2026.

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. *arXiv preprint arXiv:2305.20050*, 2023.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. Self-consistency preference optimization. *arXiv preprint arXiv:2411.04109*, 2024.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Mengjie Ren, Jie Lou, Boxi Cao, Xueru Wen, Hongyu Lin, Xianpei Han, Le Sun, Xing Yu, and Yaojie Lu. Learning from failures: Correction-oriented policy optimization with verifiable rewards. *arXiv preprint arXiv:2605.14539*, 2026.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin 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 (Volume 1: Long Papers)*, pp. 9426–9439, 2024a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

Xincheng Wei, Yifan Ding, Yoshua Li, Dongsheng Ma, Rongxiang Weng, Xunliang Cai, Wenjian Ding, and Yao Zhang. Diagevo: Diagnosis-guided self-evolution via hierarchical error memory. *arXiv preprint arXiv:2609.00768*, 2026.

Tian Xueyun, MingHua Ma, Bingbing Xu, Nuoyan Lyu, Wei Li, Heng Dong, Zheng Chu, Yuanzhuo Wang, and Huawei Shen. Learning from mistakes: Negative reasoning samples enhance out-of-domain generalization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29689–29711, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations*, volume 2026, pp. 2548–2581, 2026b.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *Advances in Neural Information Processing Systems*, 38:126546–126573, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

A Implementation and Collection Details

We provide the implementation details needed to reproduce the controlled comparisons in the main paper. Within each comparison, settings not targeted by the corresponding ablation are held fixed. In particular, Success-SFT and Full use the same initial checkpoint, frozen Solver curriculum, collected experience, decoding configuration, learning rate, batch size, and epoch count.

A.1 Training Configuration

The Challenger and Solver are initialized from the corresponding original Qwen3 checkpoint and adapted under the training budget specified in the main experimental setup. Success-SFT and Full start from the same original checkpoint and use the same verified-success targets; Full differs only by adding the failure-aware comparative term to the Solver objective. Additional optimization settings not reported in the main text are summarized in Table 7.

Table 7: Additional optimization settings used for Challenger and Solver adaptation. These settings complement the LoRA rank and training duration specified in the main experimental setup.

Setting	**Value**
LoRA alpha	32
Learning rate	1×10^{-6}
Global batch size	8
Maximum sequence length	32,768

For Full, the Solver optimizes $\mathcal{L}_{\mathrm{SFT}} + \lambda \mathcal{L}_{\mathrm{FA}}$ with $\beta = 1.0$. We use $\lambda = 0.1$ for Qwen3-4B and $\lambda = 0.2$ for Qwen3-8B. At most five distinct non-modal answer clusters are retained per accepted problem; comparative weights are proportional to their observed counts, including singleton clusters. These settings affect only the failure-aware component; the constructive SFT target is identical to that used by Success-SFT.

A.2 Curriculum Generation and Solver Collection

Bootstrap question generation is performed by the original model M_0, while the final curriculum is generated after loading the trained Challenger adapter. The Challenger adapter is then removed. All Solver experience used for Success-SFT and Full is generated by the unchanged M_0, ensuring that Challenger adaptation changes which problems are selected without changing the model that supplies Solver trajectories.

Both question generation and repeated solving use temperature 0.6, top-p 0.95, top-k 20, and a maximum generation length of 32,768 tokens. Question generation requests one problem per call. Bootstrap domains are sampled from algebra, geometry, number theory, combinatorics, and probability. Difficulty labels are sampled from *easy*, *medium*, *hard*, and *frontier* with probabilities $0.1/0.4/0.4/0.1$, respectively. The main experimental setup specifies curriculum sizes, rollout budgets, capability-matching thresholds, and the resulting modal count range.

B Prompt Templates

The curriculum prompts are intentionally lightweight. Question generation is conditioned only on a mathematical domain and target difficulty and uses no worked examples, reference solutions, or benchmark questions.

B.1 Bootstrap and Challenger Question Generation

Bootstrap generation and final Challenger generation use the same prompt template. Only the generating model changes: the bootstrap stage uses M_0, whereas final curriculum generation uses the trained Challenger adapter.

System prompt.

```
You are an expert mathematical reasoning problem setter.

Design exactly one self-contained problem whose answer is
objectively verifiable. Follow the requested mathematical
domain and target difficulty supplied in the user message.
Difficulty refers to the depth of mathematical reasoning,
not to verbose wording.

Make the problem complete, internally consistent, solvable,
and unambiguous.

Return only the problem statement inside this block:

<question>
{The complete problem statement on one or more lines}
</question>
```

User template.

```
Requested mathematical domain: {topic}
Target difficulty: {difficulty}
Generate one problem now. Follow both labels, but do not
output the labels.
```

The generator is asked for only the problem statement. Filtered bootstrap questions become Challenger training targets; they are not inserted as few-shot examples when the trained Challenger generates the final curriculum.

B.2 Solver Prompt

All repeated Solver trajectories are generated with the following system instruction:

```
Please reason step by step, and put your final answer within
\boxed{}.
```

The user message contains only the generated question. Messages are formatted with the Qwen3 chat template, and each generated trajectory is parsed into its reasoning content and final boxed answer before answer clustering.

C Staged Verification and Experience Partitioning

Verification follows the staged self-verification procedure in Self-Verified Distillation (SVD) (Lee & Liang, 2026), adapted to our answer-cluster pipeline. Here, verified successes denote modal trajectories accepted by these checks and the question-level retention criterion.

Answer clustering and pre-verification consistency. For each frozen curriculum question, M_0 produces the Solver rollout budget specified in the main experimental setup. Final boxed answers are extracted, normalized for mathematical equivalence, and merged into answer clusters. A question proceeds only if it has a unique modal cluster satisfying $\tau_{\min} < c_{\max}/K_{\mathrm{S}} \leq \tau_{\max}$, where $c_{\max}$ is the modal cluster size. This first filter removes questions with answer agreement outside the capability-matching range.

Staged verification. Trajectories in the modal cluster first pass lightweight programmatic structural checks and are then evaluated sequentially by the model-based verification stages. Following the SVD-style verifier, the stages test cycle consistency, factual and logical consistency, and overall correctness and completeness; our implementation also applies a language and formatting check.

Table 8: Answer-cluster view of one retained Solver training record. The modal cluster contains the candidate successful trajectories; one verified member supplies the constructive SFT target, a rejected modal trajectory is neutral, and non-modal clusters contribute answer-level comparative signals. Answer values are omitted because only the supervision structure is relevant to this example.

Observed outcome	Count	Use in training
Modal answer cluster	5	candidate success set
Verified modal trajectories	4	candidate SFT targets
Rejected modal trajectory	1	neutral / discarded
Competing answer cluster a_1^-	1	comparative failure signal
Competing answer cluster a_2^-	1	comparative failure signal
Competing answer cluster a_3^-	1	comparative failure signal

Verification uses the unchanged initial model M_0 without a LoRA adapter and stops as soon as a required stage fails.

Post-verification consistency and experience partition. Let v_i be the number of modal trajectories that pass all verification stages. We reapply the same capability criterion after verification and retain the problem only when $\tau_{\min} < v_i/K_{\mathrm{S}} \leq \tau_{\max}$. This second check prevents a question from entering training merely because its raw modal answer is frequent: a sufficient number of modal trajectories must remain valid after staged verification.

The retained rollouts are then partitioned according to the information they reliably provide. Verified modal trajectories form $\mathcal{E}^+$ and supply constructive targets. Modal candidates that fail verification form $\mathcal{E}^0$ and are excluded from optimization rather than treated as negative demonstrations. For retained questions, valid non-modal answer clusters form $\mathcal{E}^-$ and supply competing answers with their occurrence counts. Consequently, self-verification affects both whether a problem is usable and how its observed experience is converted into Solver supervision.

D Example of Asymmetric Experience Construction

We illustrate the data transformation with one real training record at the answer-cluster level. The example omits the underlying problem and long reasoning traces because the relevant object here is the mapping from repeated model experience to the three supervision roles.

Eight Solver rollouts produce a unique modal answer cluster with five occurrences and three non-modal answer clusters with one occurrence each. The pre-verification modal consistency is therefore $5/8$. Four of the five modal trajectories pass all verification stages, giving post-verification verified consistency of $4/8$. Both consistency values satisfy the collection criterion, so the problem is retained. The rejected modal trajectory is assigned to neutral experience rather than converted into a negative target. Table 8 summarizes the observed answer clusters and their supervision roles.

One of the four verified modal trajectories is selected by the deterministic target-selection rule used throughout the experiments and stored as the complete successful trajectory $r^+ = (h^+, a^+)$ for SFT. The three competing clusters define

$$\mathcal{A}^- = \{(a_1^-, 1), (a_2^-, 1), (a_3^-, 1)\},$$

and therefore receive equal recurrence weights,

$$\omega_1 = \omega_2 = \omega_3 = \frac{1}{3}.$$

Success-SFT uses only the complete verified trajectory. Full uses the same constructive target and additionally compares a^+ with each a_j^- under the shared verified reasoning context h^+. Complete failed reasoning traces are not directly suppressed. The example therefore makes explicit the asymmetry used throughout the method: verified success is learned as a trajectory, recurrent competing outcomes are learned comparatively at the answer level, and unverifiable experience contributes no training signal.

Table 9: Per-benchmark Qwen3-4B results for the supervision-design ablation in Table 4. Success-SFT uses one verified successful trajectory, All-positive SFT uses all verified positives, Only-FA removes constructive SFT, Trajectory-level FA applies comparative learning to complete trajectories, and Full combines the compact verified-success target with answer-level failure-aware supervision. Hard-5 and Math-9 are shaded to emphasize the aggregate metrics used for the main ablation analysis. All values are percentages.

Method	AIME24	AIME25	AIME26	HMMT25	AMC23	MATH500	Minerva	GSM8K	Olympiad	Hard-5	Math-9
Success-SFT	73.00	62.67	66.33	29.00	96.00	97.00	26.10	95.00	69.73	60.15	68.31
All-positive SFT	70.67	61.00	61.00	29.33	96.50	96.40	27.94	94.47	68.69	58.14	67.33
Only-FA	71.00	64.00	63.00	28.33	96.00	94.80	28.68	94.62	68.10	58.89	67.61
Trajectory-level FA	70.33	63.67	65.00	27.67	95.00	95.00	26.84	94.84	69.29	59.19	67.52
Full	75.00	63.67	66.67	30.67	96.50	96.80	29.41	95.07	68.99	**61.00**	**69.20**

Table 10: Per-benchmark Qwen3-4B results for the experience-construction ablation in Table 5. The variants modify staged verification, Challenger/Solver separation, Challenger specialization, or capability-aware question selection while retaining the downstream Full training objective. Hard-5 and Math-9 are shaded to emphasize the aggregate metrics used in the main analysis. All values are percentages.

Method	AIME24	AIME25	AIME26	HMMT25	AMC23	MATH500	Minerva	GSM8K	Olympiad	Hard-5	Math-9
Full	75.00	63.67	66.67	30.67	96.50	96.80	29.41	95.07	68.99	**61.00**	**69.20**
Majority-only	73.33	60.33	64.00	32.00	95.75	96.40	28.31	95.30	68.55	59.64	68.22
Coupled C./S.	72.00	60.00	66.00	27.67	95.75	95.80	26.84	95.00	68.69	58.87	67.53
No Challenger	71.33	65.00	64.67	30.33	97.25	96.00	27.57	95.00	68.55	59.98	68.41
Random selection	69.67	63.00	62.00	28.67	93.25	96.20	26.47	94.69	67.36	58.14	66.81

E Complete Ablation Results

The main paper reports ablations using the unified Hard-5 and Math-9 aggregates. For completeness, we provide the corresponding per-benchmark Qwen3-4B results below. These tables are intended as a detailed breakdown of the controlled comparisons in Section 5.4.

E.1 Supervision-Design Ablation

Table 9 provides the complete per-benchmark breakdown corresponding to the aggregate supervision-design ablation reported in the main paper.

The per-benchmark breakdown is consistent with the aggregate interpretation in the main text. Success-SFT remains the strongest constructive-only variant, while Full attains the best Hard-5 and Math-9 aggregates by adding comparative failure supervision to the same verified-success target. Individual benchmarks are not uniformly monotonic—for example, Success-SFT remains stronger than Full on MATH500 and OlympiadBench—so the aggregate comparison does not imply per-benchmark dominance. The Trajectory-level FA row provides supporting evidence for answer-level use of failure information.

E.2 Experience-Construction Ablation

Table 10 provides the corresponding per-benchmark breakdown for the experience-construction ablation.

No construction ablation improves either aggregate over Full. The detailed results nevertheless contain benchmark-level exceptions; for example, Majority-only is higher on HMMT25 and GSM8K, while No Challenger is higher on AIME25 and AMC23. The consistent aggregate degradation is therefore more informative than any single benchmark difference and supports the main-text conclusion that reliable verification, role separation, and capability-aware selection affect the utility of self-generated experience.