RLSR: REINFORCEMENT LEARNING WITH SELF-REFLECTION FOR LARGE LANGUAGE MODELS SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) can distill capacities that emerge with test-time self-reflection and incentivize large language models (LLMs) to self-improve. However, existing approaches fail to co-evolve self-reflection with direct policy optimization and instead condition the privileged self-teacher on unverified hindsight, causing self-improvement to plateau or even reverse. In this paper, we propose RLSR, a framework that optimizes and distills self-reflecting LLMs to achieve persistent self-improvement with RL. To generate endogenous learning signals, RLSR samples rollouts via grouped multi-turn self-reflection and retains only instances that achieve group-level aggregate success, grounding the privileged self-teacher in verified rather than assumed hindsight. RLSR also optimizes the policy via a dual-track co-evolutionary mechanism: a direct track internalizes breakthroughs incurred by self-reflection into the direct policy via sign-adaptive counterfactual advantage reshaping, and a reflect track assigns turn-level advantages via a log-odds belief calibration that tracks pivotal corrective transitions, actively reinforcing error correction to safeguard the self-reflection engine against degradation. These two tracks are optimized under a unified policy-gradient objective, establishing a sample-efficient self-reinforcing co-evolutionary cycle that remains stable throughout training. Across eight mathematics and code synthesis benchmarks, RLSR outperforms or matches RLVR and state-of-the-art self-distillation baselines in single-turn pass rate with preserved test-time reflection capacities.

1 INTRODUCTION

Self-improvement refers to the capacity of an autonomous system to inspect, evaluate, and iteratively refine its own policy parameters from self-generated experience (Good, 1966; Schmidhuber, 1987; Ren et al., 2026; Huang et al., 2026) and has emerged as the definitive frontier in post-training large language models (LLMs) (DeepSeek-AI, 2025; OpenAI, 2024). The primary recipe for self-improvement is Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Yu et al., 2025; Bensal et al., 2025), which optimizes an LLM against deterministic ground-truth or programmatic verifiers in verifiable domains such as mathematics and code generation. However, standard RLVR frameworks operate under coarse-grained outcome supervision: an entire reasoning trajectory spanning thousands of deliberative tokens receives a single sequence-level scalar reward. This uniform credit assignment obscures the precise causal contribution of intermediate deductions, treating critical logical pivots and catastrophic missteps with identical reinforcement (Lightman et al., 2023; Wang et al., 2026b; Lin et al., 2026).

To alleviate reward sparsity, On-Policy Self-Distillation (OPSD) (Zhao et al., 2026; Li et al., 2026) conditions a model on auxiliary training-time context to serve as the *privileged self-teacher* that provides dense, per-token likelihood supervision over the model's on-policy rollouts (Zhang et al., 2026). Nevertheless, conditioning the self-teacher on oracle ground-truth solutions leaks downstream dependencies into intermediate tokens. Enforcing a model to mimic this teacher distribution suppresses its exploratory deliberation markers, causing premature entropy collapse and degrading long-horizon reasoning (Kaur et al., 2026; Ding & Zhang, 2026). Recent efforts instead replace oracle labels with post hoc reviews (Wu et al., 2026; Yang et al., 2026b) or retrieved skill banks (Lu

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.

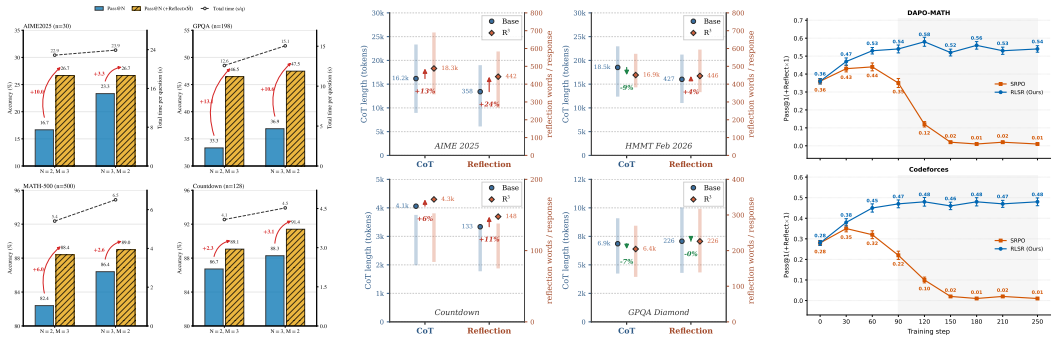


Figure 1: **Empirical motivation for RLSR.** (a) Deeper self-reflection improves pass rate but increases inference cost. (b) R^3 strengthens reflection without consistently improving the direct policy, as reflected by limited growth in CoT length and reflection markers. (c) SRPO's reflection capability degrades during training, whereas RLSR preserves it.

et al., 2026). Yet these experiential artifacts remain dependent on external repositories or analyzers, which violates the endogenous, self-reliant improvement objective of OPSD.

This paper studies the question: *How can LLMs endogenously generate dynamic, causal, and well-aligned privileged supervision for self-improvement without relying on external hints or teacher models?* We seek the answer by exploiting **Test-Time Scaling** (TTS) (Zhang et al., 2025; Snell et al., 2024). Specifically, TTS demonstrates that allocating additional computation at inference elicits latent reasoning abilities that far surpass immediate generation (Brown et al., 2024; Wei et al., 2022). *Parallel scaling* strategies use majority voting (Zuo et al., 2025) to aggregate independent reasoning traces. However, as identified by Pan et al. (2026) and Tan et al. (2025), pure consensus frequently traps the policy in self-consistent errors where identical mistakes are actively reinforced. In contrast, *sequential TTS* scales through self-reflection, in which a model receives environmental verification feedback and iteratively improves its reasoning across consecutive turns (Madaan et al., 2023; Bensal et al., 2025; Bilal et al., 2026). Figure 1(a) demonstrates that by increasing self-reflection depth, the model's cumulative pass rate persistently surges on challenging reasoning benchmarks. More importantly, when a model flips an initial or persistent failure, the resulting reaction history forms an ideal, uncorrupted source of **compute-induced privilege**: the discovered path contains both the concrete pitfalls that caused the failure and the effective interventions that led to the final success.

While translating this emergent test-time capacity into permanent parameter updates (Yu et al., 2024; Ren et al., 2026) is appealing for self-improvement, it raises unresolved challenges. On the learning objective frontier, prior efforts focus on either surrogate reflection optimization or direct-policy amortization. The former ones (Bensal et al., 2025) assume that reinforcing self-reflection turns alone intrinsically elicits reasoning deliberation and therefore leave the direct policy unamortized. However, the evidence in Figure 1(b) shows that stronger reflection does not consistently improve direct reasoning, leaving further inference turns necessary for answer refinement. The latter ones (Liu et al., 2026a) take the opposite route: they distill post hoc hindsight into the direct policy yet leave its reflective capacity untrained. As training progresses, the degraded reflective conditions weaken the privileged teacher and eventually cap the self-distillation ceiling (Figure 1(c)). On the reflection mechanism frontier, existing approaches remain predominantly confined to shallow, isolated self-reflection. Not only does single-pass correction fail to resolve complex multi-step reasoning, but reflections drawn upon failure trajectories are not verified and inevitably lead to spurious error attribution and hallucinated pitfalls—analogous to the self-consistent errors observed in pure consensus methods (Pan et al., 2026; Tan et al., 2025). Conditioning on such ungrounded hindsight injects severe off-policy noise, critically corrupting the self-teacher's supervision.

To address these problems, we propose RLSR, a Reinforcement Learning framework that optimizes and distills Self-Reflecting LLMs to achieve symbiotic self-improvement. Inspired by findings in Figure 1(a), RLSR first implements grouped self-reflection to generate rollouts with a high aggregative pass rate. It restricts training samples within the *zone of proximal development*, where at least one rollout in the group ultimately flips from failure to success after multi-turn self-reflection. This intra-group success establishes an objective empirical anchor, grounding the learning signal on hind-

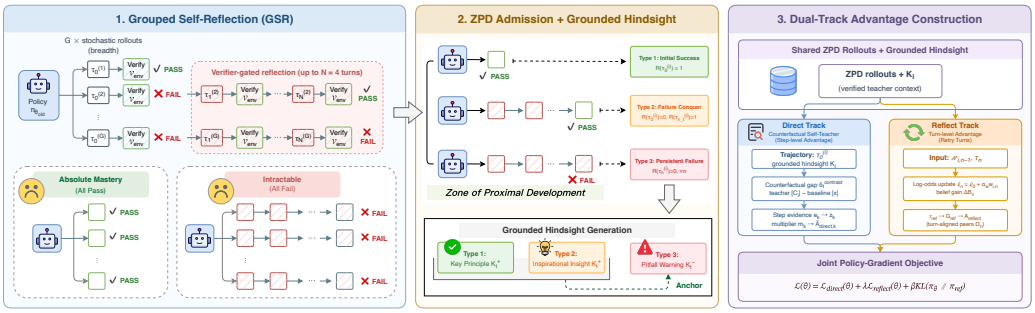


Figure 2: For each problem, GSR explores G initial rollouts and extends each by at most N retries; instances beyond ZPD are filtered out. The **direct track optimization** internalizese reflective breakthroughs, while the **reflect track optimization** preserves self-correction.

sight with supporting evidence and furnishing the dynamic, on-trajectory self-teacher with verified, compute-induced privilege. RLSR also synchronizes the capacity amortization and self-reflection optimization via a *dual-track co-evolutionary* scheme. The *Direct Track* computes the counterfactual log-likelihood gap between the direct policy and a privileged self-teacher along the identical rollout trajectory, leveraging this contrastive signal to surgically reshape the advantage of each reasoning step. Concurrently, the *Reflect Track* models multi-turn self-reflection as an episodic decision process, calibrating progress in log-odds space to detect pivotal corrective transitions (the *Crucial Flip*) and assigning turn-level advantages that reward foundational, forward-looking revisions. Optimized jointly in a unified policy gradient objective, these two tracks establish a self-reinforcing co-evolutionary cycle: the reflect track safeguards against gradient absence to ensure the self-teacher engine continually sharpens its error-correction faculties, while the direct track permanently internalizes these emergent breakthroughs into model weights, achieving sustainable, sample-efficient self-evolution. Our contributions are:

- We reconceptualize test-time scaling not merely as an inference mechanism, but as an endogenous, oracle-free generator of privileged supervision for self-improvement, preserving the autonomy and enclosure of the learning system.
- We identify the critical risk of *hallucinated privilege* and *diminishing privilege* in test-time distillation and propose a dual-track co-evolutionary RL framework to guarantee that permanent reflection faculties reinforce direct reasoning monotonically.
- On extensive experiments across mathematical reasoning and code synthesis, RLSR delivers the strongest overall results among RLVR and self-distillation baselines. Under single-pass reasoning evaluation, it leads all eight benchmarks at 4B and five of eight at 1.7B. Also, it achieves the best round-2 reflection performance across both model scales on AIME 2025 and LiveCodeBench.

2 Methodology: Reinforcement Learning with Self-Reflection

As illustrated in Figure 2, RLSR performs Grouped Self-Reflection on each problem, searching over candidate solutions along breadth (G stochastic rollouts) and depth (N verifier-gated retry turns), and restricts training to instances within the Zone of Proximal Development. The resulting rollouts feed two tracks optimized jointly under a single policy-gradient objective: the **direct track** shapes the direct policy's advantage via counterfactual self-teacher likelihood gaps on reasoning steps to internalize test-time emerging capabilities, while the **reflect track** assigns turn-level advantages to the retry turns, preventing the gradient absence that would otherwise degrade the self-teaching engine. We detail these mechanisms in the remainder of this section.

2.1 Problem Formulation

We consider complex reasoning in verifiable domains. Let $\mathcal{X}$ denote the problem space, and let $\mathcal{Y}$ be the space of token sequences. Given $x \in \mathcal{X}$, a policy π_θ generates a reasoning trajectory $\tau \in \mathcal{Y}$ ending in executable code or a parsable solution. Correctness is evaluated by a deterministic binary

verifier $\mathcal{V}_{\text{env}} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, defining the outcome reward $R(\tau) \triangleq \mathcal{V}_{\text{env}}(x, \tau)$. Unlike single-turn RLVR or static on-policy distillation (OPD), RLSR formulates learning as an episodic interaction spanning direct execution and iterative self-correction. Formally, an episode is parameterized by $(x, \tau_0, \mathcal{H}_N)$, where $\tau_0 \sim \pi_\theta(\cdot \mid x)$ is the initial zero-shot attempt, and $\mathcal{H}_N$ denotes a sequence of up to N reflection turns generated conditionally when $\mathcal{V}_{\text{env}}(x, \tau_0) = 0$ (and $\emptyset$ otherwise).

2.2 Rollout Generation and Samples Filtering

A central premise of RLSR is that a policy in a multi-turn self-reflective state constitutes an compute-induced privileged self-teacher for its own direct reasoning. However, extracting reliable supervisory signals requires overcoming hallucinated error attribution. We therefore propose *grouped self-reflection* (GSR) to address this. Specifically, given a problem x, the model first deploys stochastic group sampling at high temperature to draw a cohort of G independent initial reasoning trajectories, $\tau_0^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ for $i \in \{1, \ldots, G\}$, effectively expanding exploration breadth across semantically diverse solution paths. Whenever an initial trajectory fails the environment verifier ($R(\tau_0^{(i)}) = 0$), GSR deepens the exploration along that branch through up to N sequential turns of verifier-guided self-reflection. At each turn $n \in \{1, \ldots, N\}$, the binary environmental check $\text{fb}_{n-1} = \text{FAIL}$ is appended to construct an ongoing interaction history $\mathcal{H}_{i,n} = (x, \tau_0^{(i)}, \text{fb}_0, \ldots, \tau_n^{(i)})$, from which the policy generates a corrective continuation $\tau_n^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathcal{H}_{i,n-1})$. This iterative process proceeds until the verifier confirms correctness at turn $n_i^* \leq N$ ($R(\tau_{n_i^*}^{(i)}) = 1$) or the allotted turn budget N is exhausted, yielding an expansive search tree that exposes both successful breakthroughs and persistent deadlocks.

Unconstrained training across all explored rollouts inevitably destabilizes optimization: if all G initial trajectories solve the problem on the first attempt, the sequence-level baseline variance drops to zero ($\sigma = 0$), providing no learning gradient; conversely, if the model persistently fails across all $G \times N$ exploration steps, the absence of an objective ground-truth anchor causes the model to hallucinate incorrect causal diagnoses, reinforcing self-consistent errors. To establish an informative and stable learning regime, we filter samples outside the *Zone of Proximal Development (ZPD)* (Vygotsky, 1978), a.k.a., restricting gradient updating to problems that lie beyond the model's immediate single-turn mastery, yet remain within reach of its test-time iterative capacity:

$$\sum_{i=1}^{G} R\left(\tau_0^{(i)}\right) < G \quad \text{and} \quad \sum_{i=1}^{G} \max_{n \in \{0, \ldots, n_i^*\}} R\left(\tau_n^{(i)}\right) \geq 1. \tag{1}$$

As classified in Table 7 (Appendix E), this condition partitions a dataset into three operational regimes. Optimization is only funneled to the standard and marginal ZPD instances that provide both high causal variance and verified positive targets. Because every admitted ZPD group contains at least one verified positive rollout, these successful branches act as *objective empirical anchors*. Rather than asking the model to guess its failure reasons, RLSR prompts the model to summarize causal *hindsights* conditioned on the outcome of a trajectory: 1) **Type 1 (initial success, $R(\tau_0^{(i)}) = 1$)** synthesizes a *Key Principle* ($\mathcal{K}_i^+$) from its immediate correct deduction; 2) **Type 2 (failure conquer, $R(\tau_0^{(i)}) = 0$, $R(\tau_{n_i^*}^{(i)}) = 1$)** extracts an *Inspirational Insight* ($\mathcal{K}_i^*$) from its eventual breakthrough; and 3) **Type 3 (persistent failure, $R(\tau_n^{(i)}) = 0$ for all n)** formulates a *Pitfall Warning* ($\mathcal{K}_i^-$) by diagnosing its unresolvable loop against the group-level anchors. These hindsights parameterize dynamic, trajectory-specific self-teachers with strictly grounded privileged contexts, without external intervention.

2.3 Reflect Track: Actively Preserving the Reflection Engine

The reflect track prevents *Reflection Degradation* by optimizing multi-turn retries directly with policy gradients rather than treating them as passive distillation contexts. This preserves the model's self-correction capabilities and ensures they co-evolve with direct reasoning.

A central challenge is credit assignment: rewarding foundational scaffolding actions that facilitate eventual breakthroughs at a later turn $\bar{n}(> n)$, while properly isolating the marginal contribution of an immediate retry $\tau_n^{(i)}$ from the inherited historical context $\mathcal{H}_{i,n-1}$. To measure whether an inter-

mediate turn actively moves the reasoning trajectory toward success, we draw on Wang et al. (2026c) to calibrate empirical progress through a log-odds belief score. Let $\bar{R}_{\text{group}} = \frac{1}{G}\sum_{j=1}^{G} R(\tau_0^{(j)})$ be the group pass rate on problem x, which provides an empirical prior on problem difficulty. We clip this prior for numerical stability and map it to the log-odds space:

$$B_0 = \text{clip}\left(\bar{R}_{\text{group}},\, \epsilon_0,\, 1-\epsilon_0\right), \quad \ell_0 = \text{logit}(B_0) = \ln\left(\frac{B_0}{1-B_0}\right). \tag{2}$$

At reflection turn $n \in \{1, \ldots, n_i^*\}$, let $w_{i,n} \in [0,1]$ denote the verifier empirical win rate under state $\mathcal{H}_{i,n}$ (e.g., Avg@G). The belief score accumulates turn-level verifier evidence on top of the prior:

$$\ell_n^{(i)} = \ell_0 + \alpha_w \cdot w_{i,n}, \quad B_n^{(i)} = \sigma\left(\ell_n^{(i)}\right) = \frac{1}{1+\exp\left(-\ell_n^{(i)}\right)}, \tag{3}$$

where $\alpha_w > 0$ modulates evidence sensitivity. The *Marginal Epistemic Belief Gain* achieved by turn n is defined as:

$$\Delta B_n^{(i)} \triangleq B_n^{(i)} - B_{n-1}^{(i)} = \sigma\left(\ell_n^{(i)}\right) - \sigma\left(\ell_{n-1}^{(i)}\right). \tag{4}$$

$\Delta B_n^{(i)}$ serves as an intrinsic detector of pivotal transitions: if turn n executes a *Crucial Flip* that rectifies an intractable contradiction, $\Delta B_n^{(i)}$ spikes positively; if it introduces hallucination into an otherwise sound derivation, $\Delta B_n^{(i)}$ penalizes the policy severely. The intermediate turn-level reward at step n is then defined as:

$$r_{\text{ref}}^{(i,n)} = R\left(\tau_n^{(i)}\right) + \alpha_{\text{prog}} \Delta B_n^{(i)} - \eta_{\text{step}}, \tag{5}$$

where α_{prog} rewards belief advancement and η_{step} imposes an efficiency penalty to discourage verbose looping.

To propagate terminal recovery credit back to early foundational deductions without succumbing to myopic bias, we compute the discounted return-to-go along branch i:

$$G_{\text{ref}}^{(i,n)} = \sum_{k=n}^{n_i^*} \gamma_{\text{ref}}^{k-n} \cdot r_{\text{ref}}^{(i,k)}, \tag{6}$$

where $\gamma_{\text{ref}} \in (0,1]$ denotes the reflective temporal discount factor.

Because the active reflection depth varies across branches, normalizing returns across mismatched horizons introduces baseline skew. We resolve this by computing advantages strictly over the *Turn-Aligned Active Set* $\Omega_n \triangleq \{j \in \{1,\ldots,G\} \mid n \le n_j^*\}$ (detailed in Appendix G):

$$A_{\text{reflect}}^{(i,n)} = \begin{cases} \dfrac{G_{\text{ref}}^{(i,n)} - \mu_n}{\sigma_n + \epsilon_0}, & \text{if } |\Omega_n| \ge 2, \\ \text{clip}\left(G_{\text{ref}}^{(i,n)}, -1, 1\right), & \text{if } |\Omega_n| = 1, \end{cases} \tag{7}$$

where $\mu_n = \frac{1}{|\Omega_n|}\sum_{j\in\Omega_n} G_{\text{ref}}^{(j,n)}$ and $\sigma_n = \sqrt{\frac{1}{|\Omega_n|}\sum_{j\in\Omega_n}(G_{\text{ref}}^{(j,n)} - \mu_n)^2}$. This ensures that every reflective retry is evaluated fairly against peer trajectories confronting the identical historical failure context $\mathcal{H}_{n-1}$.

2.4 Direct Track: Step-Level Advantage Reshaping

The objective of the direct track is to compile self-reflective discoveries into direct-policy parameters such that the model solves complex problems via direct reasoning without test-time overhead.

To eliminate reward sparsity, recent on-policy distillation frameworks introduce fine-grained credit assignment by using privileged teacher log-probabilities as dense advantages. However, this conflates two decisions that should remain decoupled: the direction of the policy update and its magnitude. Because the teacher's log-ratio dictates the advantage's sign, an omniscient teacher often overrules the environment verifier - penalizing constructive exploratory backtracking that the verifier would have ultimately accepted (Kaur et al., 2026; Ding & Zhang, 2026). RLSR resolves this

pathology by keeping the verifier as the exclusive source of update direction, employing the privileged self-teacher solely to reshape step-level update magnitude via a bounded contrastive multiplier.

For an initial trajectory $\tau_0^{(i)} = (y_1, \ldots, y_{L_i}) \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, self-reflection yields one of the three grounded hindsights of Section 2.2, which we write generically as $\mathcal{K}_i \in \{\mathcal{K}_i^+, \mathcal{K}_i^*, \mathcal{K}_i^-\}$. To avoid mode collision across the diverse solution paths induced by high-temperature sampling, each privileged self-teacher is strictly bound on-trajectory by conditioning solely on that rollout's specific privileged context $\mathcal{C}_i = [x \parallel p_i \parallel \mathcal{K}_i]$, where the hint p_i is "Key Insight to Follow:" for Type 1, "Flash of Inspiration:" for Type 2, and "Avoid Pitfall:" for Type 3 (full templates in Appendix H). For each token $y_t \in \tau_0^{(i)}$, we define the *Counterfactual Log-Likelihood Gap*:

$$\delta_t^{(i),\text{contrast}} \triangleq \text{sg}\left[\log \pi_{\theta_{\text{old}}}\left(y_t \mid \mathcal{C}_i, y_{<t}\right)\right] - \text{sg}\left[\log \pi_{\theta_{\text{old}}}\left(y_t \mid \mathcal{C}, y_{<t}\right)\right], \tag{8}$$

where $\mathcal{C} = [x]$ is the baseline context and $\text{sg}[\cdot]$ denotes the stop-gradient operator. $\delta_t^{(i),\text{contrast}}$ measures how injecting reflective hindsight into the generation prefix shifts the teacher's likelihood. If token y_t constitutes a fatal logical bug, injecting the pitfall warning $\mathcal{K}_i^-$ causes the teacher's likelihood to drop sharply ($\delta_t \ll 0$). Conversely, if y_t represents a necessary lemma, its likelihood remains stable or increases ($\delta_t \geq 0$), even within an overall incorrect rollout.

Because individual tokens lack semantic autonomy in mathematical and algorithmic derivations, assigning credit at the raw token level induces high variance. We partition $\tau_0^{(i)}$ into K_i distinct reasoning steps, where step $k \in \{1, \ldots, K_i\}$ spans token indices $\mathcal{S}_k$, and aggregate token deltas into a step-level causal evidence metric:

$$e_k^{(i)} = \sum_{t \in \mathcal{S}_k} \delta_t^{(i),\text{contrast}}. \tag{9}$$

To ensure scale-invariance across problems of varying syntactic complexity, we apply *Intra-Trajectory Standardization*:

$$z_k^{(i)} = \frac{e_k^{(i)} - \mu_e^{(i)}}{\sigma_e^{(i)} + \epsilon_0}, \quad \mu_e^{(i)} = \frac{1}{K_i}\sum_{j=1}^{K_i} e_j^{(i)}, \quad \sigma_e^{(i)} = \sqrt{\frac{1}{K_i}\sum_{j=1}^{K_i}(e_j^{(i)} - \mu_e^{(i)})^2}. \tag{10}$$

Next, let $A_{\text{seq}}^{(i)} = \frac{R(\tau_0^{(i)}) - \bar{R}_{\text{group}}}{\sigma_{\text{group}} + \epsilon_0}$ denote the standard group-relative sequence advantage (as introduced by GRPO (Shao et al., 2024)). We reshape $A_{\text{seq}}^{(i)}$ into a fine-grained step advantage $\tilde{A}_{\text{direct},k}^{(i)}$ using a **sign-adaptive causal multiplier** $m_k^{(i)}$:

$$m_k^{(i)} \triangleq \text{clip}\left(1 + \text{sign}\left(A_{\text{seq}}^{(i)}\right) \cdot b \cdot z_k^{(i)},\, 1-b,\, 1+b\right), \tag{11}$$

$$\tilde{A}_{\text{direct},k}^{(i)} = A_{\text{seq}}^{(i)} \cdot \left[(1 - \lambda_{\text{resh}}) + \lambda_{\text{resh}} \cdot m_k^{(i)}\right], \tag{12}$$

where $b \in (0, 1)$ bounds the maximum relative modification, and $\lambda_{\text{resh}} \in [0, 1]$ governs the interpolation between sequence-level and step-level supervision. On a failed trajectory ($A_{\text{seq}} < 0$), sound steps receive attenuated penalties ($m_k < 1 \implies |\tilde{A}_k| < |A_{\text{seq}}|$), while fatal error steps receive amplified penalties ($m_k > 1 \implies |\tilde{A}_k| > |A_{\text{seq}}|$), achieving surgical credit assignment without optimization instability.

2.5 Unified Co-evolutionary Objective

Both tracks are executed across the identical rollouts generated through GSR, requiring no duplicate forward exploration. All sampled tokens are collated into a joint replay buffer:

$$\mathcal{D}_{\text{joint}} = \mathcal{D}_{\text{direct}} \cup \mathcal{D}_{\text{reflect}}, \tag{13}$$

where $\mathcal{D}_{\text{direct}} = \{(x, \tau_0^{(i)}) \mid i \in \{1, \ldots, G\}\}$ and $\mathcal{D}_{\text{reflect}} = \{(\mathcal{H}_{i,n-1}, \tau_n^{(i)}) \mid i \in \{1, \ldots, G\}, n \in \{1, \ldots, n_i^*\}\}$. The unified co-evolutionary objective is optimized by minimizing a clipped surrogate policy gradient loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(c,y)\sim\mathcal{D}_{\text{joint}}}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\min\left(\rho_t(\theta)\mathcal{A}_t^*,\, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right)\mathcal{A}_t^*\right)\right] + \beta \text{D}_{\text{KL}}\left(\pi_\theta \parallel \pi_{\text{ref}}\right),$$

Table 1: Main results on mathematical reasoning and code generation.

Method	AIME 24	AIME 25	MATH-500	HMMT 24	HMMT 25	HMMT 26	LiveCodeBench	BigCodeBench-Hard
Qwen3-1.7B								
Vanilla	46.46	36.67	90.60	25.62	22.92	28.98	40.54	3.38
GRPO	45.83	33.96	91.15	27.08	23.33	28.03	41.05	4.05
OPSD	42.92	35.56	89.55	23.75	21.04	27.88	40.00	4.73
OPSD+GRPO	43.96	36.89	90.10	27.29	22.08	27.47	41.19	4.39
R^3	43.54	36.89	90.50	27.71	23.13	29.49	41.16	**5.63**
SRPO	42.50	**38.22**	90.90	**28.54**	23.96	29.09	41.42	4.73
RLSD	43.75	34.67	89.50	27.08	22.92	28.69	41.68	4.84
SDPO	45.42	34.67	90.75	23.13	24.37	29.90	40.31	5.30
RLSR	**47.10**	37.00	**91.20**	27.80	**24.60**	**30.20**	**42.00**	5.50
Qwen3-4B								
Vanilla	71.88	**65.42**	94.95	35.42	38.75	40.91	49.60	10.13
GRPO	63.33	60.83	95.50	31.04	33.12	34.66	47.81	7.77
OPSD	66.67	57.11	91.65	30.83	32.50	33.13	49.91	9.45
OPSD+GRPO	63.54	61.46	95.20	30.00	32.29	33.52	48.27	5.75
R^3	64.40	61.20	94.80	30.40	33.10	33.20	48.30	5.80
SRPO	66.80	62.20	95.00	31.60	33.90	34.20	48.90	6.10
RLSD	66.46	62.44	94.40	30.21	33.75	32.73	48.52	10.40
SDPO	73.12	63.11	94.85	35.42	40.42	38.99	49.57	9.51
RLSR	**74.00**	65.09	**95.70**	**36.00**	**41.00**	**41.30**	**50.30**	**10.80**

where $\rho_t(\theta) = \frac{\pi_\theta(y_t|c,y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|c,y_{<t})}$ denotes the token-level importance sampling ratio, and the unified advantage $\mathcal{A}_t^*$ is assigned via context dispatch:

$$\mathcal{A}_t^* \triangleq \begin{cases} \mu_{\text{dir}} \cdot \tilde{A}^{(i)}_{\text{direct},\kappa(t)}, & \text{if } c = x \quad \text{(direct track token, step } \kappa(t)\text{)}, \\ \mu_{\text{ref}} \cdot A^{(i,n)}_{\text{reflect}}, & \text{if } c = \mathcal{H}_{i,n-1} \quad \text{(reflect track token, turn } n\text{)}. \end{cases} \tag{14}$$

Here, μ_{dir} and μ_{ref} satisfy $\mu_{\text{dir}} + \mu_{\text{ref}} = 1$, and π_{ref} denotes the frozen base reference model. Refer to Algorithm 1 for the overall RLSR procedure.

3 Experiments

3.1 Experimental Setup

Benchmarks. We conduct experiments using Qwen3 (Qwen Team, 2025) on two verifiable domains. For mathematical reasoning, models are trained on DAPO-MATH-17K (Yu et al., 2025) and evaluated on AIME 2024/2025 (Mathematical Association of America, 2024; 2025), HMMT 2024–2026 (Harvard-MIT Mathematics Tournament, 2024; 2025), and MATH-500 (Lightman et al., 2023) with deterministic answer matching; we report avg@16 for AIME/HMMT and avg@4 for MATH-500. For code synthesis, models are trained on CodeContests (Li et al., 2022) and evaluated with executable tests on a 240-problem post-cutoff slice of LiveCodeBench (Jain et al., 2024) and 150 BigCodeBench-Hard tasks (Zhuo et al., 2024); the primary metric is Pass@1.

Baselines. We compare RLSR with the vanilla model, GRPO (Shao et al., 2024), OPSD and OPSD+GRPO (Zhao et al., 2026; Li et al., 2026), R^3 (Bensal et al., 2025), SRPO (Liu et al., 2026a), RLSD (Yang et al., 2026a), and SDPO (Hübotter et al., 2026). These cover sparse RLVR, privileged on-policy distillation, reflection-centric RL, and hybrid objectives. All methods use the same backbone, data, verifier, prompts, per-problem sampling budget of 12 trajectories, and checkpoint-selection protocol; implementation details and the exact budget accounting are given in Appendix B.

3.2 Main Results

Overall comparison. Table 1 shows that RLSR is the most consistent method across scales and domains. At 4B, it leads every trained baseline on all eight metrics and exceeds the vanilla model on seven; at 1.7B, it leads five metrics, exceeds vanilla on all eight, and remains within 1.22 points of the best result in the remaining cells. The gains extend beyond mathematics: RLSR leads

Table 2: Track and filter ablations.

	Variant	Math			Coding
		AIME 24	AIME 25	MATH-500	LiveCodeBench
Qwen3-1.7B	RLSR (full)	47.10	37.00	91.20	42.00
	w/o direct track	43.60	35.90	90.50	41.10
	w/o reflect track	44.20	36.60	90.90	41.40
	w/o ZPD filter	45.30	35.20	90.70	41.30
Qwen3-4B	RLSR (full)	74.00	65.09	95.70	50.30
	w/o direct track	66.30	61.70	94.85	48.55
	w/o reflect track	67.50	62.70	95.15	48.95
	w/o ZPD filter	71.40	63.30	95.40	49.50

Table 3: Reflection scaling after training.

	Method	AIME 25			LiveCodeBench		
		Round=0	Round=1	Round=2	Round=0	Round=1	Round=2
Qwen3-1.7B	Vanilla	36.67	40.00	43.33	40.54	40.83	41.25
	R^3	36.89	43.33	**46.67**	41.16	42.08	42.50
	SRPO	**38.22**	38.22	38.22	41.42	41.42	41.42
	RLSR	37.00	**46.67**	**46.67**	**42.00**	**42.50**	**43.33**
Qwen3-4B	Vanilla	**65.42**	70.00	70.00	49.60	50.42	50.83
	R^3	61.20	70.00	**73.33**	48.30	50.83	52.08
	SRPO	62.20	62.20	62.20	48.90	48.90	48.90
	RLSR	65.09	**73.33**	**73.33**	**50.30**	**51.67**	**55.00**

LiveCodeBench at both scales and BigCodeBench-Hard at 4B, indicating transfer across verifiable domains without sacrificing single-pass performance. The strongest competitors are specialized: SDPO is competitive on 4B mathematics but weaker on HMMT 2026 and code, while R^3 and SRPO preserve some retry benefits without consistently improving the direct policy. The small margins in the few non-leading cells (0.13–1.22 points) further suggest that RLSR's advantage is broad robustness rather than isolated peaks.

Ablation study. Table 2 removes one RLSR component at a time under the same training and decoding protocol. Every removal lowers all four reported metrics, while the magnitudes separate their roles. Removing the direct track is most damaging: AIME 2024 drops by 3.50 points at 1.7B and 7.70 points at 4B, with a further 1.75-point loss on 4B LiveCodeBench. This confirms that reflective breakthroughs must be amortized into direct reasoning. Removing the reflect track costs 2.90 and 6.50 AIME points at 1.7B and 4B, respectively, showing that direct-policy improvement alone does not preserve the correction engine. ZPD removal has a smaller but consistent effect, including drops of 1.80 on 1.7B AIME 2025 and 2.60 on 4B AIME 2024, because unfiltered rollouts dilute the verified contrast. The three components therefore provide complementary parts of the self-improvement loop.

3.3 More Analyses

Reflection scaling after training. Table 3 evaluates the final models with and without verifier-gated retries. RLSR reaches the highest round-2 score in all four comparisons, tying R^3 on AIME 2025 and leading on LiveCodeBench at both scales. It raises AIME 2025 from 37.00 to 46.67 at 1.7B and from 65.09 to 73.33 at 4B; LiveCodeBench rises from 42.00 to 43.33 and from 50.30 to 55.00, respectively. R^3 reaches the same round-2 AIME scores but starts lower and remains behind on LiveCodeBench, whereas SRPO is flat across rounds. RLSR therefore combines a stronger direct policy with preserved retry gains: the direct track lifts single-pass performance, while the reflect track retains recovery after a failed first attempt.

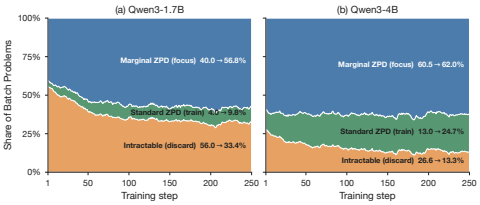


Figure 3: Composition of the DAPO-MATH-17K training batches over time.

Training dynamics. Figure 4 shows both direct and reflection-conditioned performance improving throughout training, while the gap contracts from roughly 0.4 to 0.16 for Qwen3-1.7B and from 0.6 to 0.16 for Qwen3-4B. Final reflection pass rates reach about 0.48 and 0.60, whereas OPSD and GRPO+OPD remain lower and more volatile; the 4B OPSD curve also declines near the end. Figure 3 links this trajectory to the evolving training distribution: the marginal ZPD share grows from 40.0% to 56.8% at 1.7B and from 60.5% to 62.0% at 4B, the standard-ZPD share grows from 4.0% to 9.8% and from 13.0% to 24.7%, and intractable groups shrink from 56.0% to 33.4% and from 26.6% to 13.3%. Training thus increasingly focuses on instances whose outcomes can be changed by verified reflection, while the rising standard-ZPD share indicates that part of this capability is moving into direct reasoning.

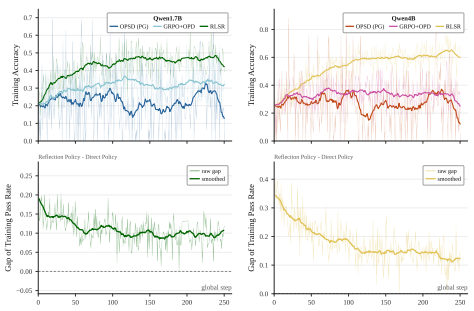


Figure 4: Comparison of training dynamics. The reflection gain is defined as reflection pass rate ($N = 2$) minus direct-policy accuracy.

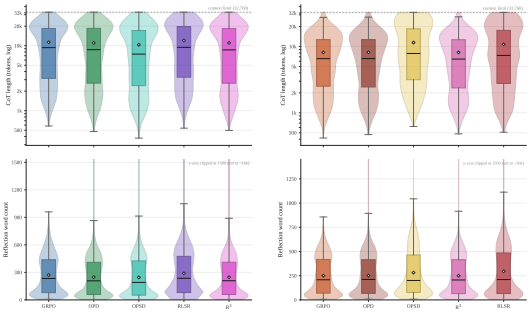


Figure 5: Post-training CoT length and reflection-marker distributions on math benchmarks. Left: Qwen3-1.7B; right: Qwen3-4B.

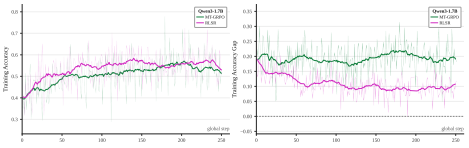


Figure 6: Reflection pass rate (left) and reflection gain (right) during training.

Why dual-track coupling matters. To explain the shrinking gap in Figure 4, we compare RLSR with a matched MT-GRPO control under the shared backbone, verifier, and rollout budget. MT-GRPO trains the direct track with standard GRPO and the reflection track with a one-retry terminal-outcome reward. This objective rewards whether a reflection episode eventually succeeds, but it neither localizes credit to the corrective turns that produced the success nor exposes their resulting insight as a learning signal for direct reasoning. RLSR closes this loop: its reflect track reinforces useful retry transitions, while its direct track uses verified reflective trajectories to reshape direct-policy advantages. Figure 6 supports this distinction: the two methods attain comparable reflection pass rates in the left panel, yet only RLSR progressively narrows the reflection–direct gap in the right panel, while MT-GRPO leaves it roughly flat or slightly larger. The matched reflection quality therefore isolates the mechanism: RLSR feeds reflective discoveries back into the direct policy, whereas MT-GRPO improves reflection as an auxiliary behavior but leaves the two tracks largely uncoupled.

Thinking mode evolution. Figure 5 shows that RLSR's gains do not come from simply generating longer chains of thought. CoT lengths remain broadly between a few thousand and tens of thousands of tokens, with medians comparable to the baselines, while reflection-marker counts shift upward and retain a heavier right tail at both scales. The pattern is clearest for Qwen3-1.7B and remains visible at 4B even when OPSD produces similarly long CoTs. RLSR therefore changes the composition, not merely the amount, of computation: additional tokens concentrate on interruption, verification, backtracking, and retry construction.

4 Conclusion

We propose RLSR, an RL framework that internalizes endogenous capacities emerging with self-reflection for LLM self-improvement. RLSR performs grouped self-reflection to explore candidate solutions for a given problem and retains training instances having rollouts within ZPD to ground privileged hindsight in verified breakthroughs. Then, the dual-track learning objective jointly preserves the self-reflection engine and transfers its discoveries into the direct policy. Across mathematical reasoning and code generation benchmarks, RLSR improves both direct reasoning and test-time refinement across model scales.

AI use statement

In this work, we used generative AI tools for code implementation, data processing, and figure generation. We have not used generative AI tools for core research idea generation or key conceptual drafting. Additionally, we used generative AI tools for LaTeX source formatting, spell checking,

and text polishing. We have reviewed all AI-assisted work. Specifically, all generated code and data processing scripts were manually inspected and thoroughly tested for correctness by the authors; created figures were verified against raw experimental data; and all polished text was carefully checked to ensure semantic fidelity and factual accuracy. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. *CoRR*, abs/2306.13649, 2024. doi: 10.48550/ARXIV.2306.13649. URL https://doi.org/10.48550/arXiv.2306.13649.

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. Reflect, retry, reward: Self-improving LLMs via reinforcement learning. *CoRR*, abs/2505.24726, 2025. doi: 10.48550/ARXIV.2505.24726. URL https://doi.org/10.48550/arXiv.2505.24726.

Ahsan Bilal, Muhammad Ahmed Mohsin, Muhammad Umer, Lena Trigg, Ali Subhan, Muhammad Ali, and Dean F. Hougen. Refining over resampling: Test-time self-correction for LLM reasoning. *CoRR*, abs/2608.05643, 2026. doi: 10.48550/ARXIV.2608.05643. URL https://doi.org/10.48550/arXiv.2608.05643.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787, 2024. doi: 10.48550/ARXIV.2407.21787. URL https://doi.org/10.48550/arXiv.2407.21787.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Yi Ding and Ruqi Zhang. Does on-policy distillation really distill? from noisy teacher to self-improvement, 2026. URL https://arxiv.org/abs/2608.31046.

Irving John Good. Speculations concerning the first ultraintelligent machine. *Advances in Computers*, 6:31–88, 1966. doi: 10.1016/S0065-2458(08)60418-0.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models. *CoRR*, abs/2306.08543, 2024. doi: 10.48550/ARXIV.2306.08543. URL https://doi.org/10.48550/arXiv.2306.08543.

Harvard-MIT Mathematics Tournament. Harvard-mit mathematics tournament (HMMT) 2024, 2024. URL https://www.hmmt.org/.

Harvard-MIT Mathematics Tournament. Harvard-mit mathematics tournament (HMMT) 2025, 2025. URL https://www.hmmt.org/.

Zixun Huang, Kishan Panaganti, Haitao Mi, and Leowei Liang. Flowbalance: Verifier-grounded self-improvement from on-policy reasoning experience, 2026. URL https://arxiv.org/abs/2609.03241.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination-free evaluation of large language models for code, 2024. URL https://livecodebench.github.io/.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *CoRR*, abs/2607.05184, 2026. doi: 10.48550/ARXIV.2607.05184. URL https://doi.org/10.48550/arXiv.2607.05184.

Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision. *CoRR*, abs/2608.06296, 2026. doi: 10.48550/ARXIV.2608.06296. URL https://doi.org/10.48550/arXiv.2608.06296.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode, 2022. URL https://deepmind.google/research/publications/competitlevel-code-generation-with-alphacode/.

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. *CoRR*, abs/2305.20050, 2023. doi: 10.48550/ARXIV.2305.20050. URL https://doi.org/10.48550/arXiv.2305.20050.

Wenze Lin, Jiale Zhao, Xitai Jiang, Songde Rao, Yining Li, Shenzhi Wang, Bingxiang He, and Gao Huang. On-policy distillation with verifiable reward. *CoRR*, abs/2608.24696, 2026. doi: 10.48550/ARXIV.2608.24696. URL https://doi.org/10.48550/arXiv.2608.24696.

Jialong Liu, Yuling Shi, Ning Yang, Xiaodong Gu, and Zuchao Li. Srpo: Self-reflective policy optimization for long-horizon reasoning, 2026a. URL https://arxiv.org/abs/2608.23493.

Ye Liu, Srijan Bansal, Bo Pang, Yang Li, Zeyu Leo Liu, Yifei Ming, Zixuan Ke, Shafiq Joty, and Semih Yavuz. Procedural memory distillation: Online reflection for self-improving language models. *CoRR*, abs/2607.01480, 2026b. doi: 10.48550/ARXIV.2607.01480. URL https://doi.org/10.48550/arXiv.2607.01480.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SKILL0: in-context agentic reinforcement learning for skill internalization. *CoRR*, abs/2604.02268, 2026. doi: 10.48550/ARXIV.2604.02268. URL https://doi.org/10.48550/arXiv.2604.02268.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, 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. *CoRR*, abs/2303.17651, 2023. doi: 10.48550/ARXIV.2303.17651. URL https://doi.org/10.48550/arXiv.2303.17651.

Mathematical Association of America. American invitational mathematics examination (AIME) 2024, 2024. URL https://www.maa.org/math-competitions.

Mathematical Association of America. American invitational mathematics examination (AIME) 2025, 2025. URL https://www.maa.org/math-competitions/.

OpenAI. OpenAI o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Teng Pan, Yuchen Yan, Zixuan Wang, Ruiqing Zhang, Guiyang Hou, Wenqi Zhang, Weiming Lu, Jun Xiao, and Yongliang Shen. Coverrl: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. *CoRR*, abs/2603.17775, 2026. doi: 10.48550/ARXIV.2603.17775. URL https://doi.org/10.48550/arXiv.2603.17775.

Qwen Team. Qwen3 technical report, 2025. URL https://github.com/QwenLM/Qwen3.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. *CoRR*, abs/2607.13104, 2026. doi: 10.48550/ARXIV.2607.13104. URL https://doi.org/10.48550/arXiv.2607.13104.

Jürgen Schmidhuber. Evolutionary principles in self-referential learning. on learning how to live: The shape of the future. *Diploma Thesis, Technische Universität München*, 1987. URL https://mediatum.ub.tum.de/813180.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/ARXIV.2408.03314. URL https://doi.org/10.48550/arXiv.2408.03314.

Hexiang Tan, Fei Sun, Sha Liu, Du Su, Qi Cao, Xin Chen, Jingang Wang, Xunliang Cai, Yuanzhuo Wang, Huawei Shen, and Xueqi Cheng. Too consistent to detect: A study of self-consistent errors in LLMs. *CoRR*, abs/2505.17656, 2025. doi: 10.48550/ARXIV.2505.17656. URL https://doi.org/10.48550/arXiv.2505.17656.

L. S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA, 1978. ISBN 9780674576285.

Aozhe Wang, Zhengxi Lu, Jianze Wang, Shangke Lv, Ying Liu, Weiming Lu, Jun Xiao, Yueting Zhuang, Hua Yang, Qianglong Chen, and Yongliang Shen. Ttpo: Test-time policy optimization, 2026a. URL https://arxiv.org/abs/2608.27448.

Chen Wang, Zhaochun Li, Jionghao Bai, Yining Zhang, Hexuan Deng, Ge Lan, and Yue Wang. Distilled reinforcement learning for LLM post-training. *CoRR*, abs/2607.17247, 2026b. doi: 10.48550/ARXIV.2607.17247. URL https://doi.org/10.48550/arXiv.2607.17247.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, Xunliang Cai, Yongliang Shen, and Yujiu Yang. Agentopsd: Recursive self-distillation for agentic reinforcement learning. *CoRR*, abs/2608.05987, 2026c. doi: 10.48550/ARXIV.2608.05987. URL https://doi.org/10.48550/arXiv.2608.05987.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. doi: 10.48550/ARXIV.2201.11903. URL https://doi.org/10.48550/arXiv.2201.11903.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: self-evolving on-policy distillation for agentic reinforcement learning. *CoRR*, abs/2607.14777, 2026. doi: 10.48550/ARXIV.2607.14777. URL https://doi.org/10.48550/arXiv.2607.14777.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR, 2026a. URL https://arxiv.org/abs/2604.03128.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: on-policy skill distillation for agentic reinforcement learning. *CoRR*, abs/2606.26790, 2026b. doi: 10.48550/ARXIV.2606.26790. URL https://doi.org/10.48550/arXiv.2606.26790.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *CoRR*, abs/2407.06023, 2024. doi: 10.48550/ARXIV.2407.06023. URL https://doi.org/10.48550/arXiv.2407.06023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

Guibin Zhang, Jiayang Lyu, Ran Sun, Xinlei Yu, Haoyu Zhao, Qibing Ren, and Shuicheng Yan. Latent on-policy self-distillation. *CoRR*, abs/2608.13040, 2026. doi: 10.48550/ARXIV.2608.13040. URL https://doi.org/10.48550/arXiv.2608.13040.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *CoRR*, abs/2503.24235, 2025. doi: 10.48550/ARXIV.2503.24235. URL https://doi.org/10.48550/arXiv.2503.24235.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *CoRR*, abs/2601.18734, 2026. doi: 10.48550/ARXIV.2601.18734. URL https://doi.org/10.48550/arXiv.2601.18734.

Terry Yue Zhuo et al. BigCodeBench: Benchmarking code generation with diverse function calls and simple instructions, 2024. URL https://arxiv.org/abs/2406.15877.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: test-time reinforcement learning. *CoRR*, abs/2504.16084, 2025. doi: 10.48550/ARXIV.2504.16084. URL https://doi.org/10.48550/arXiv.2504.16084.

A Related Work

A.1 Reinforcement Learning for Self-Improvement

The pursuit of a system improving from its own is rooted in early work on self-referential and recursive learning (Good, 1966; Schmidhuber, 1987) and has recently been revisited by self-improving agents (Ren et al., 2026). The most direct approach for self-improvement is test-time reflection: Self-Refine(Madaan et al., 2023) iteratively critiques and revises outputs, while Reflexion (Shinn et al., 2023) retains verbal feedback in an agent's episodic context. However, these methods remain limited by LLMs' capacity to diagnose their own errors and expensive inference costs. Reinforcement learning with verifiable rewards (Shao et al., 2024; Yu et al., 2025) provides a training-time alternative. However, although binary verifiers avoid costly solution annotations, their terminal rewards leave long trajectories with coarse credit assignment. Process rewarding supplies denser feedback but requires step labels or trained verifiers (Lightman et al., 2023). On-policy self-distillation instead learn from the student's own generations (Gu et al., 2024; Agarwal et al., 2024), often by comparing the same policy under ordinary and privileged contexts (Zhao et al., 2026; Li et al., 2026). Yet recent studies expose noisy-teacher, privilege-leakage, and entropy-collapse risks (Kaur et al., 2026; Ding & Zhang, 2026). Subsequent approaches seek better-grounded privilege signals, including retrieved skills (Lu et al., 2026), hindsight (Yang et al., 2026b; Wu et al., 2026; Liu et al., 2026a), and procedural memory (Liu et al., 2026b). Nevertheless, these learning signals are generated from additional analyzers, which deviate from the intention of self-improvement. RLSR instead optimizes and distills the self-reflection capacity through online policy learning, thereby achieving persistent, causal improvement by the model itself.

A.2 Test-Time Scaling

Test-time scaling studies how additional inference compute can improve predictions from a fixed model (Zhang et al., 2025). Parallel scaling first elicits intermediate reasoning through chain-of-thought prompts (Wei et al., 2022), then improves coverage through repeated sampling (Brown et al., 2024) and marginalizes independent chains by majority voting (Wang et al., 2023). Despite prevalence, recent analyses of self-consistent errors (Tan et al., 2025; Pan et al., 2026) show that consensus objectives can collapse diversity and reinforce correlated mistakes. Sequential scaling follows another axis by iteratively critiquing and correcting a trajectory, as advocated in Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023). Recent work on distilling System 2 reasoning into System 1 responses shows that deliberate computation can be amortized into faster responses (Yu et al., 2024). Still, ordinary distillation primarily transfers final direct behavior and can leave the reflection-conditioned mode underexercised. At the same time, transductive test-time updates such as TTRL (Zuo et al., 2025) and TTPO (Wang et al., 2026a) are tied to the encountered unlabeled instances. Unlike these inference-only or consensus-based approaches, RLSR treats breadth-depth test-time search as a grounded supervision generator. GSR jointly explores parallel rollouts and recursive corrections, ZPD filtering concentrates learning on instances where extra compute actually changes the verifier outcome, the direct track converts temporary successes into persistent first-pass capability, and the reflect track continues optimizing multi-turn recovery. RLSR therefore bridges TTS and RL for self-improvement: it neither equates majority opinion with truth nor leaves scaling gains trapped at inference time.

B Experimental Setup and Implementation Details

B.1 Models, Data, and Contamination Audit

All methods are initialized from the same Qwen3-1.7B or Qwen3-4B backbone (Qwen Team, 2025). For mathematics, we use DAPO-MATH-17K (Yu et al., 2025) as the shared training corpus. For code, we use a public Codeforces-derived CodeContests training split (Li et al., 2022), retaining only Python problems with executable public test cases and a verifiable canonical input–output interface. Both corpora are filtered to remove prompts whose final answers or test suites can be shortcut by template matching.

We perform two contamination checks against every evaluation benchmark: (i) normalized 8-gram overlap between training prompts and benchmark statements, and (ii) exact matching of normalized

Table 4: **Benchmark protocol.** All evaluations use fixed prompt templates, maximum lengths, and decoding parameters across methods.

Domain	Benchmark	Split	Primary Metric	Protocol
Math	AIME 2024	test	avg@16 accuracy	answer-equivalence verifier; 95% paired-bootstrap CI
Math	AIME 2025	test	avg@16 accuracy	answer-equivalence verifier; 95% paired-bootstrap CI
Math	MATH-500	test	avg@4 accuracy	answer-equivalence verifier and subject breakdown
Math	HMMT 2024	test	avg@16 accuracy	answer-equivalence verifier; 95% paired-bootstrap CI
Math	HMMT 2025	test	avg@16 accuracy	answer-equivalence verifier; 95% paired-bootstrap CI
Code	LiveCodeBench	post-cutoff slice	Pass@1	executable unit tests; platform/difficulty-stratified 240-problem slice
Code	BigCodeBench-Hard	official hard set	Pass@1	executable unit tests; 150-task user-facing and challenging subset

final answers, problem IDs, and test-case hashes. Any candidate match above the audit threshold is removed from training before any model is trained. We also report the post-filter corpus size and the number of removed candidates in the released training manifest.

B.2 Evaluation Protocol

Table 4 summarizes the benchmark splits and metrics. For mathematical reasoning, we extract the final boxed answer and score it with a deterministic equivalence checker; no LLM judge is used for the main metric. AIME and HMMT are evaluated with avg@16, while MATH-500 uses avg@4 to balance cost and variance. For LiveCodeBench, we use the official code-generation scenario and report executable Pass@1 on a fixed 240-problem post-cutoff slice stratified by platform and difficulty. For BigCodeBench-Hard, we use the official 150-task executable subset and report Pass@1.

B.3 Training Environment and Compute Budget

All experiments are conducted on a single node equipped with 8× NVIDIA A800 GPUs. RLSR is trained for 250 outer iterations using the rollout configuration of Table 8 ($G = 4$ stochastic rollouts per problem, a training-time reflection depth of $N = 2$ verifier-gated retry turns, and a maximum of 16,384 generated tokens per turn) and the optimization settings of Table 9.

B.4 Baseline Implementation and Budget Accounting

All methods use the same backbone, tokenizer, reward verifier, prompt template, maximum generation length, batch size, optimizer schedule, and total training iterations. Unless otherwise stated, we train for 250 outer iterations with 256 problems per iteration, and every baseline draws 12 rollouts per problem, matching RLSR's maximum per-problem search budget. RLSR spends that budget differently, trading breadth for depth: it draws $G = 4$ initial rollouts and then applies up to $N = 2$ verifier-gated retry turns to each failed branch, so a single problem consumes at most $G(1+N) = 12$ trajectories—exactly the budget granted to each baseline. Because retries are gated on failure and terminate at the first verified success, RLSR's average trajectory count per problem is at most 12 and in practice below the baselines', which always instantiate the full group; the equal-budget comparison is therefore conservative with respect to RLSR. We adopt this allocation deliberately: it is depth, not breadth, that turns an initial failure into a verified success, and only such a success can serve as the empirical anchor admitted by the ZPD filter and consumed by both tracks. Moreover, only instances admitted by the ZPD criterion (Eq. 1) are optimized, so RLSR also performs fewer gradient updates per sampled problem than baselines that update on every rollout; Appendix B.5 reports the

Table 5: **Baseline configuration.** All methods are retrained on the same domain-specific corpus and evaluated with the same verifier.

Method	Training Signal	Privileged Context
GRPO	sparse verifier reward	none
OPSD	dense on-policy self-distillation	ground-truth solution / final answer
OPSD+GRPO	dense on-policy self-distillation plus sparse outcome reward	ground-truth solution / final answer
R^3	verifier-gated reflection-token reward	none
SRPO	reverse-KL reward from a reflection-augmented self-teacher	self-generated reflection patch
RLSD	verifier direction with self-distilled magnitude	none
SDPO	dense self-distillation from a successful sibling rollout	successful prior rollout

corresponding total generated tokens, verifier calls, and wall-clock time. OPSD+GRPO keeps the OPSD objective but adds the verifier's outcome reward, so it requires no separate teacher-training stage; all methods are therefore trained under the same single-stage budget. For all methods, we sweep learning rate, KL coefficient, and rollout temperature on a common validation split.

Table 5 summarizes method-specific access. OPSD receives ground-truth privileged context only during training. OPSD+GRPO inherits the ground-truth privileged context of OPSD and additionally receives the sparse verifiable outcome reward; the two signals are combined during training only. R^3 rewards successful reflection tokens. SRPO is reimplemented without its original SFT stage. For Qwen3-1.7B, the R^3 and SRPO entries correspond to the TASKREW-Qwen3-1.7B-step200 and TASKREW-Qwen3-1.7B-step250 checkpoints, respectively. RLSD retains verifier-derived update direction and uses self-distillation only for update magnitude. SDPO uses a successful prior rollout as privileged context.

B.5 Reflection Inference and Cost Accounting

We report two inference regimes. *First-attempt* evaluation forbids reflection and measures the amortized single-pass policy. *RLSR + reflection* evaluation permits up to two verifier-gated retries and reports final accuracy, reflection gain, average reflection turns, average generated tokens, and verifier calls. This separates durable policy improvement from inference-time test-time scaling. For all methods, we report total generated tokens and verifier calls during training, not only outer optimization steps. When wall-clock cost is reported, it includes rollout, verifier execution, teacher scoring, and optimization time.

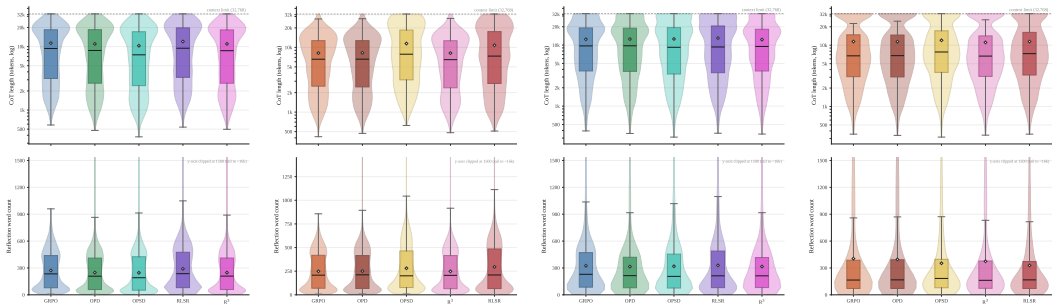
B.6 Statistical Analysis

Each main result is averaged over five independent training seeds. The seed changes prompt order, rollout sampling, and gradient-noise realization, while the backbone, data, and hyperparameter grid are fixed. Confidence intervals are computed by paired bootstrap over 10,000 resamples of per-instance scores. For paired comparisons against GRPO and the strongest distillation baseline, we report two-sided p-values. Because AIME and HMMT contain few problems, we use $\mathrm{avg@16}$ and report confidence intervals rather than interpreting a single Pass@1 point estimate.

C More Results

C.1 CoT and Reflection-Token Distributions

To examine how post-training changes generation shape, we compare the output-length statistics of GRPO, OPD, OPSD, RLSR, and R^3 on both Qwen3-1.7B and Qwen3-4B. For each model-training-



(a) Qwen3-1.7B, Math (b) Qwen3-4B, Math (c) Qwen3-1.7B, Code (d) Qwen3-4B, Code

Figure 7: **CoT and reflection-token distributions after training.** Each panel shows violin-box plots over generations from one model scale and training domain. Within each panel, the upper row reports CoT length in tokens (log scale) and the lower row reports reflection-marker count. The reflection axis is clipped at 1,500 for display; the right tail extends to roughly 16k.

domain pair, we collect generations under the same decoding budget used for evaluation, count reflection markers separately from ordinary chain-of-thought, and report two quantities: CoT length in tokens on a log scale and reflection-marker count. Table 6 lists the marker lexicon, split into a high-precision core and an exploratory tier, used for this count. The violin plots show the sample density; the overlaid boxes show the interquartile range, median, and mean. The dashed line in the top panels marks the 32,768-token context limit. For readability, the reflection axis is clipped at 1,500 markers, although the observed right tail extends to roughly 16k.

Table 6: **Reflection-marker lexicon.** Matches are case-insensitive, word-boundary anchored, and allow flexible whitespace. The *Reflective* tier collects interruption, backtracking, and error-correction triggers that are unambiguously reflective; the *Exploratory* tier collects markers that signal a retry or an alternative route but also occur in ordinary chain-of-thought. The two tiers form the counting lexicon (21 entries). Generic high-frequency discourse markers and quantifiers are excluded, because in mathematical and algorithmic derivations they attach to ordinary reasoning and would inflate the count.

Tier	Markers
Reflective	wait, hmm, oh, ah, hold on, on second thought, reconsider, recheck, double-check, alternatively, actually, mistake, wrong, incorrect
Exploratory / verification	alternative, another, other, again, maybe, check, verify

Figure 7 shows that RLSR does not obtain its gains by uniformly producing longer chains. The bulk of CoT lengths remains between a few thousand and tens of thousands of tokens across algorithms, and RLSR's median CoT length is comparable to the baselines. In contrast, RLSR exhibits a heavier right tail in reflection-marker count, especially on the code-training panels, while the 4B models show the same qualitative pattern more clearly than the 1.7B models. This indicates that the additional computation induced by RLSR is concentrated in active error diagnosis and retry construction rather than indiscriminate verbose reasoning.

C.2 Pass@N Reflection Scaling across Model Scales

Figure 8 expands Figure 1(a) by reporting the same test-time scaling experiment at both model scales (Qwen3-1.7B on the left, Qwen3-4B on the right) on all four benchmarks, under two budgets, $(N{=}2, M{=}3)$ and $(N{=}3, M{=}2)$. Within each panel, plain Pass@N is contrasted with the reflection-augmented Pass@N, the bar labels report the absolute accuracies and their difference, and the secondary axis reports the wall-clock time per question. Reflection improves accuracy on every benchmark and at both model scales: the gains are $+10.0$ and $+3.3$ points on AIME2025 and $+13.1$ and $+10.6$ points on GPQA for the 1.7B model, versus $+10.0$ and $+13.3$ on AIME2025 and $+14.6$ and $+15.7$ on GPQA for the 4B model; on the two near-saturated benchmarks they shrink

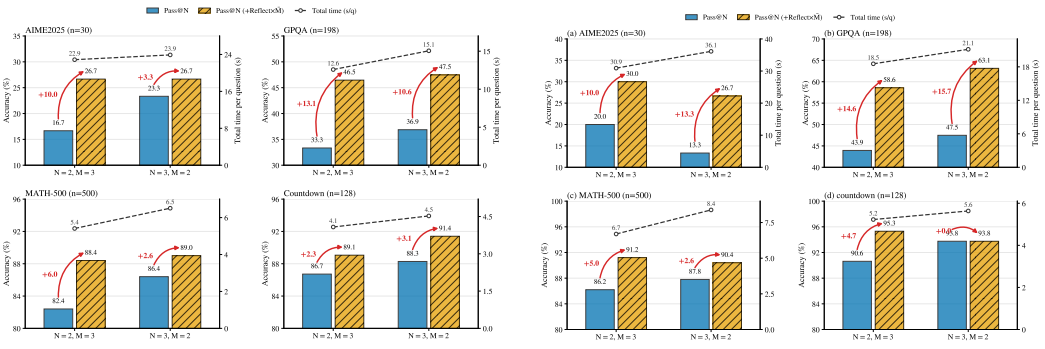


Figure 8: Reflection-augmented Pass@N calculated by Qwen3-1.7B (Left) and Qwen3-4B (right).

to +6.0 and +2.6 (MATH-500) and +2.3 and +3.1 (Countdown) for 1.7B, and +5.0 and +2.6 (MATH-500) and +4.7 and +0.0 (Countdown) for 4B. The ordering mirrors the main-text observation: reflection contributes most where the unaugmented pass rate is lowest and the headroom is largest, and its benefit disappears once the model approaches saturation—the 4B model already solves 93.8% of one Countdown setting, where reflection adds nothing. Two caveats are worth noting. First, AIME2025 contains only 30 problems, so a single problem corresponds to 3.3 accuracy points and the reported gains are granular at that resolution; the GPQA (n=198), MATH-500 (n=500), and Countdown (n=128) panels are correspondingly more stable. Second, the wall-clock axis shows that the accuracy gains are not simply bought with inference compute: the per-question cost increases only modestly, and on the near-saturated benchmarks it stays essentially flat or even decreases.

D Algorithmic Implementation Details

Algorithm 1 provides the complete operational workflow of the RLSR framework. All hyperparameters referenced by the algorithm are consolidated in Table 8 (Appendix F).

E ZPD Instance Partitioning Taxonomy

Table 7 restates the partition induced by the ZPD admission criterion of Eq. 1. Instances in the Absolute Mastery and Intractable regimes are skipped or discarded, whereas the Standard and Marginal ZPD regimes supply intra-group contrast and verified positive anchors, respectively.

Table 7: Taxonomy of Candidate Instance Partitioning in RLSR via the ZPD.

Cognitive Region	**Initial Pass (G)**	**Recursive Retry ($G \times N$)**	**Training Status**	**Theoretical Rationale**
Absolute Mastery	All Correct (G/G)	N/A (Bypassed)	✗ Skip	Zero variance ($\sigma = 0$); yields zero gradient.
Intractable	All Fail ($0/G$)	All Retries Fail	✗ Discard	Lacks objective anchor; prevents error reinforcement.
Standard ZPD	Mixed ($0 < k < G$)	Partial Recovery	✓ **Train**	Natural intra-group positive/negative contrast.
Marginal ZPD	All Fail ($0/G$)	At least 1 Success	✓ **Focus**	Compute breaks deadlocks; peak learning signal.

F Hyperparameter Configurations

Table 8 consolidates the complete set of hyperparameters of the RLSR algorithm, organized by the module in which they act: the GSR rollout configuration (Section 2.2), the reflect track (Section 2.3), the direct track (Section 2.4), and the unified co-evolutionary objective (Section 2.5). Values in **bold** are explicitly specified in the main text and are adopted verbatim; all remaining entries report the default empirical values shared across our mathematical reasoning and code synthesis experiments. Table 9 separately reports the general training and optimization hyperparameters, which are independent of the algorithmic design and kept identical across all compared methods and benchmarks. The final results are robust to moderate perturbations of the default values in Table 8, and no benchmark-specific re-tuning was performed.

Table 8: **Hyperparameters of the RLSR Algorithm.** Values in **bold** are explicitly specified in the main text; all remaining values correspond to the default empirical configuration used in our experiments.

Hyperparameter	Symbol	Value
GSR Rollout Configuration (Section 2.2)		
Group size (search breadth)	G	4
Reflection depth (max retry turns)	N	2
Sampling temperature	T	1.0
Max generation length per turn	L_{gen}	16,384
Track 1: reflect track (Section 2.3)		
Prior clip constant	ϵ_0	10^{-2}
Evidence sensitivity	α_w	4.0
Belief-progress coefficient	α_{prog}	0.5
Step efficiency penalty	η_{step}	0.05
Reflective discount factor	γ_{ref}	0.95
Singleton-advantage clip range	—	$[-1, 1]$
Track 2: direct track (Section 2.4)		
Step bound	b	**0.2**
Reshaping interpolation	λ_{resh}	0.5
Minimum step length	$L_{\min}$	**15**
Maximum step length	$L_{\max}$	**128**
Unified Co-evolutionary Objective (Section 2.5)		
Direct-track advantage weight	μ_{dir}	**0.8**
Reflective-track advantage weight	μ_{ref}	**0.2**
PPO clip ratio	ϵ	0.2
KL penalty coefficient	β	10^{-3}

G Semantic Reasoning Unit Partitioning: Protocols and Signal Specifications

In Section 2.4, the direct track relies on step-level evidential aggregation over discrete reasoning units $\mathcal{S}_k \subset \{1, \ldots, |\tau_0^{(i)}|\}$. Assigning credit at the raw token level incurs severe variance and syntactic noise, whereas assigning credit solely at the sequence level yields coarse, non-informative feedback. Therefore, RLSR establishes a deterministic, multi-tier partitioning protocol that segments long reasoning chains into logically cohesive *Reasoning Units*.

G.1 Formal Segmentation Protocol

Let a generated trajectory $\tau = (y_1, y_2, \ldots, y_L)$ consist of L tokens. A valid semantic partition of τ into K reasoning units is defined as an ordered sequence of non-overlapping, contiguous token index sets $\mathcal{P} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$ satisfying:

$$\bigcup_{k=1}^{K} \mathcal{S}_k = \{1, 2, \ldots, L\}, \quad \text{and} \quad \mathcal{S}_j \cap \mathcal{S}_k = \emptyset \quad (\forall j \neq k), \tag{15}$$

where each $\mathcal{S}_k = \{t \in \mathbb{N} \mid L_{k-1} < t \leq L_k\}$ is demarcated by a boundary index L_k triggered by an authorized delimiter signal.

To prevent pathological segmentation (e.g., trivial splits on single punctuation marks or monolithic multi-step paragraphs), the partitioning engine enforces two structural constraints:

Table 9: **General Training and Optimization Hyperparameters.** These settings are independent of the RLSR algorithm and are kept identical across all compared methods and benchmarks.

Hyperparameter	Value
Optimizer	AdamW
Optimizer betas (β_1, β_2)	$(0.9,\ 0.95)$
Weight decay	0.0
Peak learning rate	10^{-6}
Learning-rate schedule	Cosine decay to 10% of the peak learning rate
Warmup	Linear warmup over the first 5 iterations (ratio 0.02)
Gradient-norm clipping	1.0 (global norm)
Training precision	bf16 mixed precision
Batch size	256 problems per iteration
Policy updates per batch	1 (strictly on-policy)
Total training iterations	250
Rollouts per problem	12 (matched across all methods)

1. **Minimum Step Length Constraint ($L_{\min} = 15$ tokens):** If a detected delimiter occurs at index t such that $t - L_{k-1} < L_{\min}$, the boundary signal is suppressed, and the tokens are merged into the ongoing unit. This prevents isolated mathematical symbols (e.g., "$= 5$.") from forming degenerated steps.
2. **Maximum Step Length Threshold ($L_{\max} = 128$ tokens):** If an ongoing step exceeds $L_{\max}$ without encountering a primary structural delimiter, secondary weak boundaries (e.g., single sentence terminators followed by whitespace '$.\texttt{s+}$') are activated to enforce soft segmentation, avoiding oversized credit-assignment windows.

G.2 Partition Signals across Domains

The segmentation signals are hierarchical and domain-adaptive, encompassing:

- **Syntactic & Structural Delimiters:** Standardized markdown headers, numbered enumerations, and double newlines that signify a natural pause or layout shift in reasoning.
- **Deductive Connectives (Mathematical Reasoning):** Natural language and symbolic transition operators indicating logical derivation, case analysis, or lemma conclusion.
- **Syntactic AST Boundaries (Code Generation):** Statement-level terminators, function/class definitions, conditional branching headers, and loop closures in code blocks.
- **Deliberative Backtracking Markers:** Metacognitive phrasing where the model pauses, detects inconsistencies, or shifts deductive direction (critical for isolating error-correction steps).

Table 10 provides the comprehensive catalog of regular expression triggers, semantic connectives, and their functional roles across different reasoning domains.

H Prompt Templates for Grounded Teacher Spectrum

Table 11 specifies the exact prompt formats employed during the GSR synthesis phase to extract hindsights ($\mathcal{K}_i$) and condition the counterfactual self-teachers.

Table 10: **Comprehensive Taxonomy of Boundary Signals for Reasoning Unit Partitioning.** Delimiter patterns are evaluated hierarchically; high-priority structural triggers override low-priority sentence boundaries.

Signal Category	Syntactic Trigger Pattern (Regex)	Functional / Semantic Role
Mathematical Reasoning		
Stepwise Enumeration	`\n\s*(Step\s*\d+|[0-9]+\.)[:\s]`	Explicit structural division of distinct proof phases.
Deductive Transition	`(?<=[.?!])\s+(Therefore|Thus|Hence|Consequently|So,)`	Connective signaling an intermediate lemma or deduction.
Display Equation Break	`(\$\$[\s\S]*?\$\$|\\[[\s\S]*?\\])\s*\n`	Self-contained mathematical derivation blocks.
Hypothesis / Case Split	`\n\s*(Case\s*\d+|Claim\s*\d*|Subgoal|Assume that)`	Branching points for case-by-case exploration.
Code Generation		
Block Definition	`\n\s*(def|class)\s+[a-zA-Z_0-9]+`	Function or class interface specification.
Control Flow Branch	`\n\s*(if|elif|else|for|while|try|except):`	Local algorithmic decisions and loop boundaries.
Code Chunk Closure	`\n\s*(return|yield|raise|break|continue)\b`	Terminal execution point of a localized logic block.
Block Wrapper	`\n\s*```(python|cpp|java)?\n`	Boundary between natural-language thought and execution code.
Deliberative CoT		
Metacognitive Pause	`(?<=[.?!])\s*(Wait|Hold on|Alternatively|Let me double check)`	Point of hesitation indicating active self-verification.
Error Detection	`(?<=[.?!])\s*(However,|Notice that|This leads to a contradiction)`	Recognition of a flawed premise or invalid deduction.
Backtracking Pivot	`(?<=[.?!])\s*(Instead,|Let's reconsider|Restarting with)`	Crucial pivot discarding a previous invalid avenue.
Universal Fallback		
Paragraph Boundary	`\n\s*\n+`	General typographic paragraph termination.
Length-Gated Sentence	`(?<=[.?!])\s+(?=[A-Z])`	Soft boundary bounding overly verbose steps (active when $\|\mathcal{S}_k\| > L_{\max}$).

Algorithm 1 The Complete RLSR Training Procedure

Require: Training dataset $\mathcal{D}_{\text{train}}$, environment verifier $\mathcal{V}_{\text{env}}$, policy π_θ, reference policy π_{ref}, group size G, max reflection turns N, step bound b, mixture weights $\mu_{\text{dir}}, \mu_{\text{ref}}$.

for each training iteration **do**
  Sample a mini-batch of problems $\mathcal{B} \subset \mathcal{D}_{\text{train}}$.
  Initialize replay buffers: $\mathcal{D}_{\text{direct}} \leftarrow \emptyset$, $\mathcal{D}_{\text{reflect}} \leftarrow \emptyset$.
  for all $x \in \mathcal{B}$ **do**
    // Phase 1: Search Breadth (Stochastic Group Rollouts)
    Sample G initial trajectories: $\tau_0^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x), \quad i \in \{1, \ldots, G\}$.
    Evaluate outcome rewards: $R_0^{(i)} \leftarrow \mathcal{V}_{\text{env}}(x, \tau_0^{(i)})$.
    // Phase 2: Search Depth (Verifier-Gated Recursive Reflection)
    for $i = 1$ **to** G **do**
      if $R_0^{(i)} = 0$ **then**
        Initialize history: $\mathcal{H}_{i,0} \leftarrow (x, \tau_0^{(i)}, \text{FAIL})$.
        for $n = 1$ **to** N **do**
          Sample retry: $\tau_n^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathcal{H}_{i,n-1})$.
          Check correctness: $R_n^{(i)} \leftarrow \mathcal{V}_{\text{env}}(x, \tau_n^{(i)})$.
          Compute epistemic belief gain: $\Delta B_n^{(i)} \leftarrow \sigma(\ell_0 + \alpha_w w_{i,n}) - \sigma(\ell_0 + \alpha_w w_{i,n-1})$.
          Update history: $\mathcal{H}_{i,n} \leftarrow (\mathcal{H}_{i,n-1}, \tau_n^{(i)}, \text{fb}_n)$.
          if $R_n^{(i)} = 1$ **then**
            Record terminal turn $n_i^* \leftarrow n$; **break**
          end if
        end for
      end if
    end for
    // Phase 3: Cognitive Region Gating (ZPD Check)
    if $\sum_i R_0^{(i)} == G$ **or** $\max_{i,n} R_n^{(i)} == 0$ **then**
      continue {Bypass Absolute Mastery and Intractable instances}
    end if
    // Phase 4: Teacher Spectrum Synthesis
    Identify group anchor trajectory τ^* where $\mathcal{V}_{\text{env}}(x, \tau^*) = 1$.
    Extract grounded hindsights $\{\mathcal{K}_i^*\}_{i=1}^G$ for Types 1, 2, and 3 using prompt templates in Appendix H.
    // Phase 5: Reflect Track Advantage Construction
    for all active reflection steps (i, n) **do**
      Compute discounted return $G_{\text{ref}}^{(i,n)} = \sum_{k=n}^{n_i^*} \gamma_{\text{ref}}^{k-n} r_{\text{ref}}^{(i,k)}$.
      Standardize across turn-aligned peers Ω_n to obtain $A_{\text{reflect}}^{(i,n)}$.
      Append to buffer: $\mathcal{D}_{\text{reflect}} \leftarrow \mathcal{D}_{\text{reflect}} \cup \{(\mathcal{H}_{i,n-1}, \tau_n^{(i)}, A_{\text{reflect}}^{(i,n)})\}$.
    end for
    // Phase 6: Direct Track Step Causal Reshaping
    for $i = 1$ **to** G **do**
      Construct counterfactual teacher context $\mathcal{C}_i = [x \parallel p_i \parallel \mathcal{K}_i]$.
      for each token t in $\tau_0^{(i)}$ **do**
        Compute counterfactual likelihood delta: $\delta_t^{(i),\text{contrast}} = \log \pi_{\theta_{\text{old}}}(y_t \mid \mathcal{C}_i, y_{<t}) - \log \pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})$.
      end for
      Aggregate over semantic steps $\mathcal{S}_k$: $e_k^{(i)} = \sum_{t \in \mathcal{S}_k} \delta_t^{(i),\text{contrast}}$.
      Standardize step evidence $z_k^{(i)}$ and compute multiplier $m_k^{(i)}$.
      Reshape sequence advantage: $\tilde{A}_{\text{direct},k}^{(i)} = A_{\text{seq}}^{(i)} \cdot [(1 - \lambda_{\text{resh}}) + \lambda_{\text{resh}} m_k^{(i)}]$.
      Append to buffer: $\mathcal{D}_{\text{direct}} \leftarrow \mathcal{D}_{\text{direct}} \cup \{(x, \tau_0^{(i)}, \{\tilde{A}_{\text{direct},k}^{(i)}\})\}$.
    end for
  end for
  // Phase 7: Joint Co-evolutionary Policy Update
  Construct $\mathcal{D}_{\text{joint}} \leftarrow \mathcal{D}_{\text{direct}} \cup \mathcal{D}_{\text{reflect}}$.
  Update policy parameters θ by descending along $\nabla_\theta \mathcal{L}_{\text{RLSR}}(\theta)$ via AdamW.
end for

Table 11: **System Prompt Templates for hindsight Generation and Self-Teacher Contexts.**

Interaction Category	Prompt Template / In-Context Formulation
Type 1: Initial Success	`[Instruction]`: You solved the problem correctly on your first attempt. Analyze your reasoning trace and state the key mathematical invariance or algorithmic principle that guaranteed correctness. Be concise and generalizable.
Teacher Context $\mathcal{C}_i^+$	`Problem:` $\{x\}$ `Key Insight to Follow:` $\{\mathcal{K}_i^+\}$ `Solution:`
Type 2: Crucial Flip	`[Instruction]`: Your initial attempt failed, but you resolved the error at retry turn $\{n^*\}$. Compare your original erroneous reasoning with your verified solution. Explicitly identify: (1) The deceptive assumption in the initial trace, and (2) The pivotal correction that unblocked the path.
Teacher Context $\mathcal{C}_i^*$	`Problem:` $\{x\}$ `Flash of Inspiration (Correction):` $\{\mathcal{K}_i^*\}$ `Solution:`
Type 3: Persistent Failure	`[Instruction]`: Your attempts failed repeatedly across multiple turns. In contrast to the verified group solution $\{\tau^*\}$, diagnose the persistent dead-end or cyclic trap that derailed your logic. Formulate a direct warning specifying what path must NOT be taken.
Teacher Context $\mathcal{C}_i^-$	`Problem:` $\{x\}$ `Avoid Pitfall:` $\{\mathcal{K}_i^-\}$ `Solution:`