LEARNING WHAT TO PRACTICE: DIAGNOSIS-GUIDED SELF-EVOLUTION FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-play supports the self-evolution of language models, but solver performance can plateau or decline across rounds without guidance. Existing unguided methods typically use difficulty, learnability, or diversity signals to keep questions challenging and varied, without identifying which unresolved reasoning weaknesses to target. Existing guided methods rely on external task resources such as human examples, document corpora, or specified difficulty targets. We introduce **DiagEvo**, which guides question generation using the solver's failure history from self-play, without external task resources. Its diagnostician extracts recurring error causes and stores them in an error-cause memory. The memory groups related causes under skill nodes and tracks each as *Active* or *Mastered* according to self-consistency on targeted questions. The challenger uses these states and recurrence counts to balance cause-targeted generation with free exploration. *Double-confidence filtering* retains intermediate-difficulty questions only when the most common solver answer has a clear vote lead. With the default 4B diagnostician, DiagEvo outperforms all baselines in mean accuracy across nine benchmarks for each solver: Qwen3-4B, Qwen3-8B, and OctoThinker-8B. On Qwen3-8B, DiagEvo reaches 72.3% mean accuracy across five mathematical reasoning benchmarks, 4.5 percentage points above R-Zero. Its overall mean accuracy across nine benchmarks is 57.4%, 3.5 percentage points above SPICE. Ablations show that mixed generation, memory-state updates with cross-state stitching, and double-confidence filtering contribute to these gains.

1 INTRODUCTION

Recursive self-improvement (RSI) studies whether AI can improve how later systems are trained (Chi et al., 2026). Recent work examines agents that design training algorithms or revise training data strategies (Chi et al., 2026; Meng et al., 2026). Self-evolution is a related direction that aims to improve models with limited ongoing human supervision (Tao et al., 2024). Self-play provides a practical approach (Huang et al., 2026; Zhao et al., 2025).

In self-play, the challenger generates questions and the solver learns to answer them. Alternating updates let questions evolve with the solver across rounds. Questions must stay near the solver's *competence boundary*, exposing weaknesses while remaining learnable. Overly easy questions offer little improvement, while overly difficult ones produce unstable responses and weak training signals (Bae et al., 2026).

Existing self-play methods address this challenge in two ways. Label-free methods use solver responses to adjust difficulty (Huang et al., 2026; Zhao et al., 2025). Different questions can expose the same error cause, so difficulty alone does not identify what the solver should practice next. Questions can grow longer without targeting these causes, and performance can stop improving (Figure 1(a) and (b)) (Huang et al., 2026; Yu et al., 2025). Training on unreliable majority-vote pseudo-labels from difficult or unclear questions can also reinforce errors across rounds (Fan et al., 2026).

Guided self-play uses external task resources to stabilize training. R-Few uses human-annotated examples to guide question generation (Yu et al., 2025), while SPICE combines a document corpus with solver feedback to control question difficulty (Liu et al., 2025a). DARC uses explicit difficulty

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.

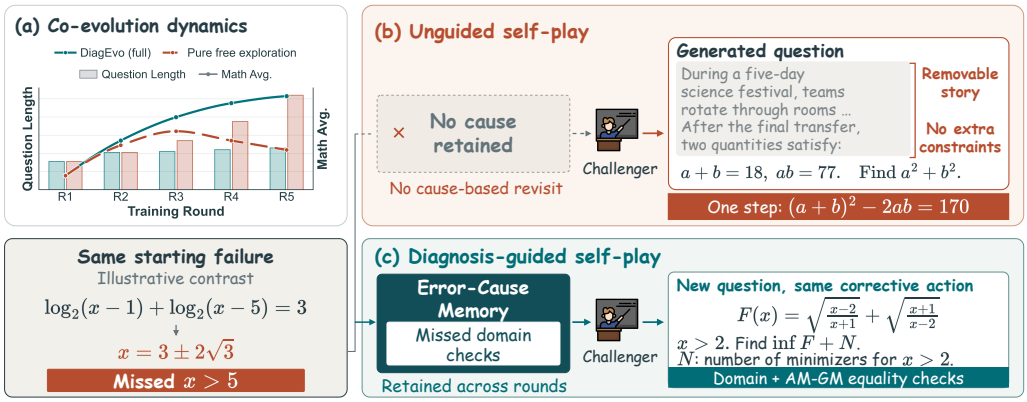


Figure 1: Co-evolution with and without error-cause memory. (a) Question length (bars) and mathematical average (lines). (b, c) An illustrative contrast from the same failure: unguided self-play adds story text without new constraints, while DiagEvo retains the error cause to guide questions requiring domain and AM-GM equality checks.

targets, external documents, and a document-informed privileged teacher (Fan et al., 2026). Continued improvement thus depends on externally supplied task data or difficulty targets.

The solver's failure history offers a curriculum signal for future questions. Prior work on LLM agents stores past interactions in memory to guide later decisions or model updates (Shinn et al., 2023; Zhao et al., 2024; Liu et al., 2025b; Ouyang et al., 2026; Cao et al., 2026; Xia et al., 2026a). For the same question, we compare a failed trajectory that disagrees with the pseudo-label with one that agrees. Their reasoning differences reveal transferable error causes to target in new questions. The memory uses recurrence counts to prioritize causes. Self-consistency on targeted questions indicates when to reduce direct targeting as the solver evolves. The same error cause links diagnosis, question generation, and progress tracking across questions and rounds.

We propose **DiagEvo**, a diagnosis-guided self-play framework (Figure 1(c)). A lightweight external LLM diagnostician extracts transferable error causes from solver trajectories produced during self-play and stores them in an error-cause memory. Each cause is *Active* while directly targeted for generation and becomes *Mastered* when the solver reaches high self-consistency on related questions. The challenger uses this memory to combine cause-targeted generation with free exploration. *Double-confidence filtering* removes questions whose two leading answers have similar vote shares. The solver learns from the retained questions and pseudo-labels, and its failed training trajectories update the memory for the next round.

The main contributions are as follows.

1. We show that self-play can use its own training experience to guide what the solver practices next. Recurring failures reveal reasoning weaknesses that question difficulty alone does not identify.
2. We introduce DiagEvo, a diagnosis-guided self-play framework. Its error-cause memory connects failure diagnosis with question generation, adapting the curriculum to the solver's evolving weaknesses without external task resources.
3. With the default 4B diagnostician, DiagEvo achieves the best overall performance among the compared methods on Qwen3-4B, Qwen3-8B, and OctoThinker-8B. On Qwen3-8B, it reaches a 72.3% mathematical reasoning average, 4.5 points above R-Zero, and exceeds DARC overall.

2 Method

DiagEvo uses failed solver trajectories to guide question generation in the next round (Figure 2). The challenger uses memory to generate curriculum candidates, and double-confidence filtering selects

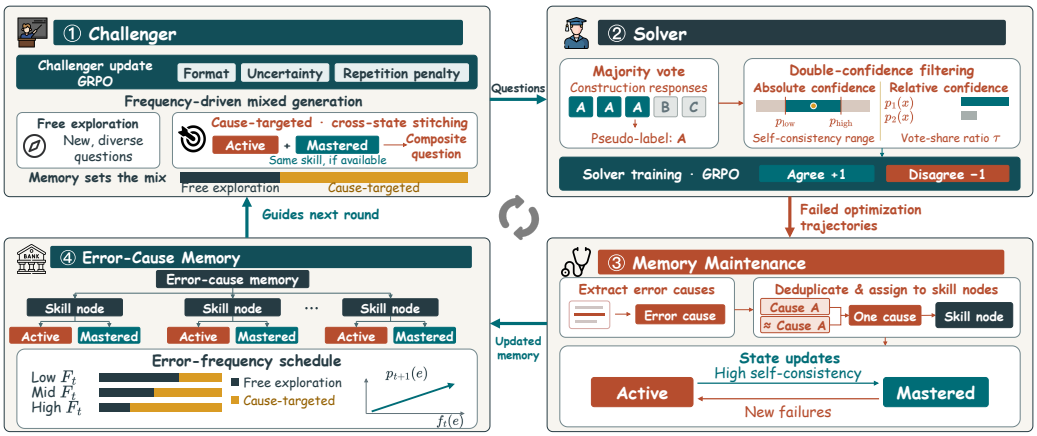


Figure 2: One DiagEvo round. (1) GRPO updates the challenger, which mixes free exploration with cause-targeted generation. (2) Double-confidence filtering selects question and pseudo-label pairs for solver training. (3) Memory maintenance compares failed optimization trajectories with construction references, then deduplicates, assigns, and updates the extracted causes. (4) The memory groups causes under skill nodes; their states and frequencies schedule the next round.

question and pseudo-label pairs for solver training. The solver's failed trajectories then update the memory.

The challenger and solver update in turn, with one frozen while the other updates. Only the challenger receives the memory.

2.1 Diagnosis-Guided Question Generation

Cause-targeted generation focuses on *Active* causes in the previous round's memory. Free exploration maintains broad question coverage and can reveal unrecorded error causes.

Frequency-Driven Mixed Generation. Let $\mathcal{M}_t$ denote the error-cause memory committed after all updates in round t (Section 2.3). For a cause e, an active episode lasts from entry into *Active* to promotion to *Mastered*. Its frequency $f_t(e)$ counts diagnosed failures within the episode and resets to zero at promotion. The total $F_t = \sum_{e \in \mathcal{E}_t} f_t(e)$ sums frequencies across current active episodes.

Round 1 uses free exploration because the memory is empty. Its failure count F_1 is the fixed reference for $z_t = F_t/F_1$. The odds of cause-targeted generation relative to free exploration grow linearly with z_t, giving the next round's exploration probability and cause distribution:

$$\frac{1-\varepsilon_{t+1}}{\varepsilon_{t+1}} = \frac{z_t}{k} = \frac{F_t}{kF_1}, \implies \varepsilon_{t+1} = \frac{1}{1+z_t/k} = \frac{F_1}{F_1+F_t/k}, \qquad p_{t+1}(e) = \frac{f_t(e)}{F_t}. \tag{1}$$

The parameter k balances the two modes, with equal probability at $z_t = k$. The distribution $p_{t+1}(e)$ samples *Active* causes in proportion to their active-episode frequencies. If $F_t = 0$, we use only free exploration, set $\varepsilon_{t+1} = 1$, and do not compute p_{t+1}. More failures under *Active* causes increase cause-targeted generation. Without reactivation by new failures, promotion reduces F_t and shifts probability back to free exploration.

Let $q_\theta^{\text{free}}(x)$ denote the challenger's free-exploration distribution and $q_\theta^{\text{tar}}(x \mid e, \mathcal{M}_t)$ its distribution conditioned on an *Active* cause e. For $F_t > 0$, the complete question distribution is

$$q_{t+1}(x \mid \mathcal{M}_t) = \varepsilon_{t+1} q_\theta^{\text{free}}(x) + (1-\varepsilon_{t+1}) \sum_{e \in \mathcal{E}_t} p_{t+1}(e)\, q_\theta^{\text{tar}}(x \mid e, \mathcal{M}_t). \tag{2}$$

The mixture combines free exploration with targeting a cause sampled by frequency.

Within cause-targeted generation, cross-state stitching pairs the sampled *Active* cause with a uniformly sampled *Mastered* cause from the same skill node. The question requires skills from both causes. If no *Mastered* sibling exists, the prompt uses only the *Active* cause.

Challenger Update and Quality Reward. The current challenger θ_t samples an update batch $\mathcal{Q}_{t+1}^{\text{chal}}$ from the mixed policy. Using the frozen solver's answers, GRPO updates the challenger to θ_{t+1} with the R-Zero reward (Huang et al., 2026). This reward favors questions near 50% self-consistency and penalizes repetition. The updated challenger samples a fresh pool $\mathcal{Q}_{t+1}^{\text{cur}}$ under the same memory schedule for filtering and solver training (Section 2.2).

2.2 Solver Optimization with Double-Confidence Filtering

Training on $\mathcal{Q}_t^{\text{cur}}$ requires reliable pseudo-labels. With ground-truth answers, intermediate success rates provide stronger RLVR learning signals (Bae et al., 2026). Generated questions lack these labels, so DiagEvo estimates difficulty from solver self-consistency. The absolute confidence constraint selects intermediate-difficulty questions but ignores the second most common answer. High-conflict questions have similar vote shares for their two leading answers. Their pseudo-labels can change across response samples. Training on them can reinforce errors across rounds. Double-confidence filtering compares these vote shares through a relative confidence constraint.

Dataset Construction and Double-Confidence Filtering. From this subsection onward, $\mathcal{Q}_t^{\text{cur}}$ contains the candidates for solver round t. The solver samples N construction responses per candidate x. Their majority answer gives the pseudo-label $\hat{y}(x)$. Their vote distribution determines filtering and provides evidence for state promotion (Section 2.3).

Among construction responses, $p_1(x)$ and $p_2(x)$ are the vote shares of the majority and second most common answers, respectively. If no second answer exists, set $p_2(x) = 0$. The self-consistency score $p(x) = p_1(x)$ measures answer agreement, not correctness. We retain questions satisfying both confidence constraints:

$$\mathcal{D}_{\text{train}} = \left\{ x \;\middle|\; p_{\text{low}} \le p(x) \le p_{\text{high}}, \quad p_1(x) \ge \tau p_2(x) \right\}. \tag{3}$$

The bounds p_{low} and p_{high} set the self-consistency range, while τ sets the required ratio of the two leading vote shares. For each retained question, we store one construction response that agrees with $\hat{y}(x)$ as the reference trajectory for diagnosis. GRPO (Shao et al., 2024) samples G separate optimization responses $\{r_j^{\text{opt}}\}_{j=1}^{G}$. Each receives $+1$ for agreement with $\hat{y}(x)$ and -1 otherwise. These responses provide the policy-gradient signal. Their failed trajectories enter diagnosis (Section 2.3).

2.3 Error-Cause Memory and State Updates

The memory groups related error causes under skill nodes. Cause states and frequencies guide question generation as the solver changes. We represent this two-level memory as

$$\mathcal{M}_t = (\mathcal{V}_t, \mathcal{E}_t, a_t), \qquad v = (d_v, \mathbf{z}_v), \qquad e = (d_e, \mathbf{h}_e, s_t(e), f_t(e)). \tag{4}$$

Here, $\mathcal{V}_t$ and $\mathcal{E}_t$ contain skill nodes and error causes, respectively. The mapping $a_t : \mathcal{E}_t \to \mathcal{V}_t$ assigns each cause to one skill node. Each node v stores a description d_v and an embedding $\mathbf{z}_v$. Causes store descriptions d_e, embeddings $\mathbf{h}_e$, curriculum states $s_t(e) \in \{\textit{Active}, \textit{Mastered}\}$, and active-episode frequencies $f_t(e)$ (Section 2.1).

Each round extracts and deduplicates causes, assigns new causes to skill nodes, and consolidates redundant nodes. The selected Qwen3-Instruct-2507 diagnostician makes all LLM-based semantic decisions using self-play data and the resulting memory. An embedding model supports retrieval within this memory. Both are frozen external models and receive no external task resources. Extraction runs independently for each failure. Later memory updates are sequential.

Comparative Error-Cause Extraction. After solver optimization in round t, diagnosis uses failed optimization trajectories from questions retained by Equation 3. For each question x, we pair a failed trajectory r^- with its stored construction reference r^+. This reference remains available even when all optimization responses disagree with $\hat{y}(x)$. The diagnostician compares the pair to find their earliest reasoning difference:

$$e^{\text{new}} = D_{\text{ext}}\big(x, \hat{y}(x), r^-, r^+\big), \qquad r^- \not\sim \hat{y}(x), \quad r^+ \sim \hat{y}(x). \tag{5}$$

Here, $\sim$ denotes answer agreement. The extracted cause excludes question-specific values and surface form. This comparison identifies a relative reasoning difference without verifying ground-truth correctness.

The set $\mathcal{E}^{\text{known}}(x)$ contains the causes supplied during generation: none for free exploration, the sampled *Active* cause for single-cause generation, or both the *Active* and stitched *Mastered* causes for cross-state generation. The diagnostician compares the earliest reasoning difference with every supplied cause and assigns each failure to at most one match. A confirmed match adds one to the cause's failure count $\Delta_t(e)$ and reactivates it if *Mastered*. If none matches, Equation 5 produces a new candidate cause for memory matching.

Deduplication, Assignment, and Consolidation. Memory maintenance retrieves similar stored causes by embedding. The diagnostician confirms matches despite wording differences, assigns each observation to the matched cause, and adds one to $\Delta_t(e)$. It assigns unmatched candidates to a retrieved or new skill node. New causes start as *Active* with frequency one. At round end, node embeddings retrieve potentially redundant skill nodes. The diagnostician confirms merges while preserving cause states and frequencies. Appendix I provides prompt templates and retrieval and routing details.

Joint State and Frequency Update. An *Active* cause is eligible for direct targeting. *Mastered* is a curriculum state based on high self-consistency, not verified correctness. Causes in this state remain available for cross-state stitching.

For each cause present at the start of round t, we first apply promotion to its state from round $t-1$. The set $\mathcal{P}_t(e) \subseteq \mathcal{Q}_t^{\text{cur}}$ contains all candidates targeting an *Active* cause e, including those removed by filtering. Their mean self-consistency gives the promotion score:

$$\text{acc}_t(e) = \frac{1}{|\mathcal{P}_t(e)|} \sum_{x \in \mathcal{P}_t(e)} p_1(x), \qquad |\mathcal{P}_t(e)| > 0. \tag{6}$$

Using the promotion threshold θ_{up}, we record the provisional state and frequency as

$$(\tilde{s}_t(e), \tilde{f}_t(e)) = \begin{cases} (\textit{Mastered}, 0), & s_{t-1}(e) = \textit{Active}, \quad |\mathcal{P}_t(e)| > 0, \quad \text{acc}_t(e) \geq \theta_{\text{up}}, \\ (s_{t-1}(e), f_{t-1}(e)), & \text{otherwise.} \end{cases} \tag{7}$$

New optimization failures can reactivate a cause promoted earlier in the same round.

After solver optimization, $\Delta_t(e)$ counts failed optimization trajectories from retained questions assigned to e. Each failure is assigned to at most one cause through recurrence attribution or memory matching. For existing causes, the committed update is

$$(s_t(e), f_t(e)) = \begin{cases} (\tilde{s}_t(e), \tilde{f}_t(e)), & \Delta_t(e) = 0, \\ (\textit{Active}, \tilde{f}_t(e) + \Delta_t(e)), & \Delta_t(e) > 0, \quad \tilde{s}_t(e) = \textit{Active}, \\ (\textit{Active}, \Delta_t(e)), & \Delta_t(e) > 0, \quad \tilde{s}_t(e) = \textit{Mastered}. \end{cases} \tag{8}$$

Active causes accumulate failures within their current episode. Reactivated causes count only new failures, excluding those from closed episodes. The committed states and frequencies determine F_t, ε_{t+1}, and p_{t+1} for the next round.

3 Experiments and Analysis

We conduct experiments to answer three questions. Section 3.1 describes the models, baselines, benchmarks, and evaluation protocol. How does DiagEvo compare with prior methods across solvers, diagnostician scales, and reasoning domains (Section 3.2)? How do mixed generation, memory-state updates with cross-state stitching, and double-confidence filtering affect performance (Section 3.3)? How do solver performance, error-cause memory, and pseudo-label quality change across rounds (Section 3.4)? A case study traces how a diagnosed failure shapes later questions and solver behavior (Section 3.5).

3.1 Experimental Setup

Models. We evaluate DiagEvo on Qwen3-4B-Base, Qwen3-8B-Base, and OctoThinker-8B-Hybrid-Base. For each solver, we test Qwen3-4B-Instruct-2507, Qwen3-30B-A3B-Instruct-2507, and Qwen3-235B-A22B-Instruct-2507 as diagnosticians.

Table 1: Reasoning accuracy (%). DiagEvo reports mean ± standard deviation over three runs; aggregate columns show means only. 'Diag.' denotes the Qwen3-Instruct-2507 diagnostician. Blue shades mark the top three means per column and block, darkest first. Bold and underlining mark first and second, including ties. † denotes results reported in DARC (Fan et al., 2026); others are ours.

Method (Diagnostician)	**Mathematical Reasoning**						**General Reasoning**					**Overall**
	AMC	Minerva	MATH-500	GSM8K	Olympiad	**Avg.**	MMLU-Pro	SuperGPQA	GPQA-D	BBEH	**Avg.**	
Qwen3-4B-Base												
Base	47.7	41.5	68.4	72.4	35.0	53.0	51.4	25.4	26.3	8.2	27.8	41.8
R-Zero	47.8	50.4	74.4	90.1	40.3	60.6	53.9	27.7	35.9	10.2	31.9	47.9
Absolute Zero†	50.0	41.9	76.2	89.3	41.5	59.8	52.6	27.1	35.3	8.3	30.8	46.9
SPICE†	50.9	55.5	77.9	91.9	41.9	63.6	56.5	28.3	37.9	11.3	33.5	50.2
R-Few (1%)†	52.7	52.1	77.8	92.3	42.4	63.5	55.9	29.4	35.4	11.2	33.0	49.9
DARC†	60.3	57.7	77.6	91.9	45.8	66.7	56.9	29.2	38.9	11.2	34.1	52.2
DiagEvo **(Diag. 4B)**	62.2 ±0.3	60.7 ±1.0	78.2 ±0.2	92.0 ±0.3	48.3 ±0.5	68.3	**57.0** ±0.4	30.3 ±0.4	41.2 ±1.2	**11.5** ±0.4	35.0	53.5
DiagEvo **(Diag. 30B-A3B)**	62.5 ±0.5	61.4 ±0.6	78.5 ±0.6	91.9 ±0.2	50.1 ±0.7	68.9	56.9 ±0.1	**30.7** ±0.5	**42.1** ±0.8	11.3 ±0.3	**35.3**	53.9
DiagEvo **(Diag. 235B-A22B)**	**63.0** ±0.4	**61.6** ±0.8	**78.7** ±0.4	**92.4** ±0.3	**50.7** ±0.4	**69.3**	**57.0** ±0.6	30.6 ±0.4	41.9 ±0.9	11.3 ±0.3	35.2	**54.1**
Qwen3-8B-Base												
Base	61.7	50.0	74.2	91.1	40.1	63.4	58.1	30.2	33.8	10.5	33.2	50.0
R-Zero	62.7	59.6	80.8	92.3	43.6	67.8	61.7	31.8	39.9	11.4	36.2	53.8
Absolute Zero†	62.5	52.9	76.6	92.0	47.8	66.4	62.5	33.5	36.8	10.8	35.9	52.8
SPICE†	60.9	55.2	81.4	93.8	48.0	67.9	61.0	32.4	40.4	12.1	36.5	53.9
R-Few (1%)†	69.3	59.6	81.6	94.0	44.0	69.7	62.8	32.7	40.4	11.8	36.9	55.1
DARC†	68.9	61.4	83.0	94.0	48.4	71.1	62.3	32.8	44.4	11.8	37.8	56.3
DiagEvo **(Diag. 4B)**	69.6 ±0.3	63.2 ±0.6	83.3 ±0.3	93.8 ±0.2	51.4 ±0.7	72.3	62.8 ±0.7	33.8 ±0.6	46.0 ±1.3	12.4 ±0.5	38.8	57.4
DiagEvo **(Diag. 30B-A3B)**	69.8 ±0.2	64.1 ±0.4	83.5 ±0.7	93.8 ±0.5	53.5 ±0.4	72.9	63.0 ±0.2	33.9 ±0.3	47.3 ±0.6	**12.6** ±0.1	39.2	57.9
DiagEvo **(Diag. 235B-A22B)**	**70.0** ±0.5	**64.8** ±0.9	**83.8** ±0.9	**94.1** ±0.4	**53.8** ±0.6	**73.3**	**63.1** ±0.4	**34.0** ±0.4	**47.5** ±0.9	12.5 ±0.3	**39.3**	**58.2**
OctoThinker-8B-Hybrid-Base												
Base	27.7	21.7	44.6	68.4	16.9	35.9	14.9	11.3	15.7	0.7	10.7	24.7
R-Zero	32.2	32.7	58.0	84.7	22.4	46.0	37.1	17.7	21.2	7.7	20.9	34.9
Absolute Zero†	32.5	34.9	56.8	87.0	25.6	47.4	31.4	18.8	27.8	5.0	20.8	35.5
SPICE†	**35.2**	40.8	58.4	87.3	25.6	49.5	41.3	19.9	29.8	7.2	24.6	38.4
DARC†	31.9	43.0	62.4	88.0	30.7	51.2	43.8	22.3	32.3	10.8	27.3	40.6
DiagEvo **(Diag. 4B)**	33.7 ±0.7	45.2 ±1.1	62.8 ±0.6	88.2 ±0.2	33.6 ±0.8	52.7	44.3 ±0.5	23.1 ±0.4	35.0 ±1.1	11.2 ±0.2	28.4	41.9
DiagEvo **(Diag. 30B-A3B)**	33.7 ±0.4	46.4 ±0.6	63.4 ±0.3	88.2 ±0.4	36.0 ±0.7	53.5	**44.8** ±0.2	23.5 ±0.2	35.4 ±0.5	**11.3** ±0.2	**28.8**	42.5
DiagEvo **(Diag. 235B-A22B)**	33.9 ±0.4	**47.2** ±0.8	**63.7** ±0.4	**88.5** ±0.5	**36.4** ±0.4	**53.9**	44.6 ±0.3	**23.6** ±0.3	**35.5** ±0.8	**11.3** ±0.4	**28.8**	**42.7**

Baselines. We compare DiagEvo with the untrained base model and five self-evolution methods: R-Zero (Huang et al., 2026), Absolute Zero (Zhao et al., 2025), SPICE (Liu et al., 2025a), R-Few (Yu et al., 2025), and DARC (Fan et al., 2026). R-Zero, Absolute Zero, and DiagEvo are label-free: they use no external task resources during training, while external models are allowed. SPICE, R-Few, and DARC use external task resources. We evaluate the base models and reproduce R-Zero. Other baseline scores come from DARC (Fan et al., 2026) under the same evaluation protocol.

Evaluation Data and Protocol. We evaluate mathematical reasoning on MATH-500 (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021), OlympiadBench (He et al., 2024), Minerva Math (Lewkowycz et al., 2022), and AMC (zwhe99, 2024). General reasoning benchmarks are MMLU-Pro (Wang et al., 2024), SuperGPQA (Du et al., 2025), GPQA-Diamond (Rein et al., 2024), and BBEH (Kazemi et al., 2025). We report accuracy, using mean@32 for AMC. Each DiagEvo run uses the checkpoint with the highest validation accuracy for all nine benchmarks. We report the mean and standard deviation over three independent runs. Appendix F provides evaluation details and baseline score validation.

3.2 Main Results

Comparison with Prior Methods. With the default 4B diagnostician, DiagEvo ranks first overall on all three solvers, exceeding every baseline by at least 1.1 points. On Qwen3-8B, its mathematical and general averages reach 72.3% and 38.8%. These exceed R-Zero by 4.5 and 2.6 points, and DARC by 1.2 and 1.0 points. DiagEvo trains without external task resources, unlike DARC.

Table 2: Ablations on Qwen3-8B-Base. DiagEvo (full) is the shared reference, using $\tau = 1.6$ for double-confidence filtering. Each variant changes one component. Subscripts show changes from full in percentage points.

	Question generation policy			Memory-state updates and cross-state stitching			Pseudo-label filtering		Reference
Metric	Frozen challenger	Pure free exploration	Pure cause-targeted generation	No state updates	No cross-state stitching	Random-pair stitching	No filtering	Absolute confidence constraint only	**DiagEvo (full)**
Math Avg.	68.5 $_{-3.8}$	69.5 $_{-2.8}$	70.1 $_{-2.2}$	70.1 $_{-2.2}$	70.8 $_{-1.5}$	71.3 $_{-1.0}$	69.4 $_{-2.9}$	70.9 $_{-1.4}$	**72.3**
General Avg.	36.9 $_{-1.9}$	37.2 $_{-1.6}$	37.6 $_{-1.2}$	37.1 $_{-1.7}$	37.5 $_{-1.3}$	37.9 $_{-0.9}$	36.7 $_{-2.1}$	37.5 $_{-1.3}$	**38.8**

Overall scores are 53.5% on Qwen3-4B and 41.9% on OctoThinker-8B, both 1.3 points above DARC.

Effect of Diagnostician Scale. Larger diagnosticians produce consistent but modest gains in mathematical reasoning. Scaling from 4B to 235B-A22B raises the mathematical average by 1.0 point on both Qwen3 solvers and 1.2 points on OctoThinker-8B. The largest gains appear on harder mathematical benchmarks. On OlympiadBench, the gains are 2.4, 2.4, and 2.8 points across the three solvers.

General averages change by only 0.2 to 0.5 points, with no gain on some benchmarks. Diagnostician scale is therefore not the main source of DiagEvo's gains.

Cross-Domain Generalization. The solver trains only on mathematical questions, so gains on general-reasoning benchmarks measure cross-domain transfer. With the default diagnostician, general averages rise from 33.2% to 38.8% on Qwen3-8B and from 27.8% to 35.0% on Qwen3-4B. OctoThinker-8B improves from 10.7% to 28.4%. These gains span different solver scales and architectures.

Training Time Breakdown. With the default 4B diagnostician, diagnosis and memory maintenance account for 5.7% of total training time. Table 6 reports the breakdown by component.

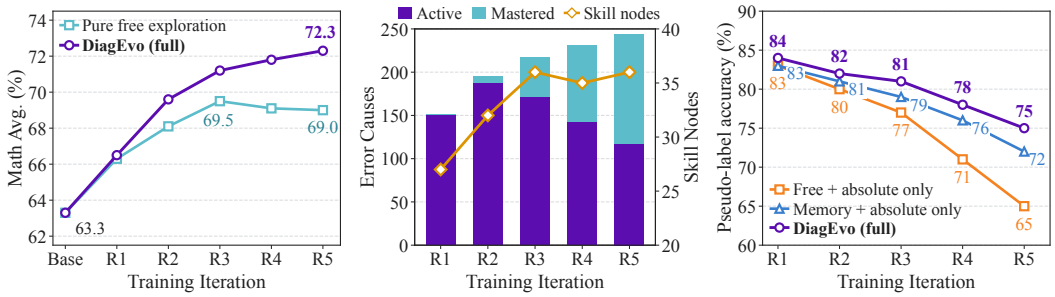
3.3 Ablation Studies

We evaluate the mixed generation policy, memory-state updates with cross-state stitching, and double-confidence filtering on Qwen3-8B-Base (Table 2). Component ablations keep the remaining hyperparameters at the full method's defaults. Each run uses the checkpoint with the highest validation accuracy. DiagEvo (full) and pure free exploration are each trained for seven rounds.

Question Generation Policy. Freezing the challenger reduces the mathematical average by 3.8 points, the largest drop in this group. The full policy outperforms both single-mode variants, showing that free exploration and cause-targeted generation are complementary. Its frequency-driven schedule exceeds the best tested fixed ratio by 1.1 points in mathematical average and 1.2 in general average (Table 8).

Memory-State Updates and Cross-State Stitching. Without state updates, all causes remain *Active*, so cross-state stitching is unavailable. Enabling promotion and reactivation without stitching raises mathematical and general averages by 0.7 and 0.4 points, respectively. Adding cross-state stitching yields a further 1.5 and 1.3 points. Selecting the *Mastered* cause from the same skill node also improves the mathematical average by 1.0 point over random-pair stitching.

Pseudo-Label Filtering. The absolute confidence constraint improves both averages over no filtering. Adding the relative confidence constraint raises the mathematical average by a further 1.4 points, showing that both constraints contribute. The mathematical average varies by at most 1.0 point across the tested values of either k or τ (Appendix G).



(a) Per-round performance (b) Memory lifecycle (c) Pseudo-label accuracy

Figure 3: Co-evolution on Qwen3-8B-Base. (a) Mathematical averages, both with double-confidence filtering. (b) Counts of *Active* and *Mastered* causes and skill nodes. (c) Pseudo-label accuracy (%) after each round's training on 200 shared questions. Starting with pure free exploration and only the absolute confidence constraint, the three settings add memory, then the relative confidence constraint.

3.4 Co-Evolution Dynamics

Solver Performance and Memory Dynamics. DiagEvo's mathematical average improves in each of the first five rounds, reaching 72.3% (Figure 3(a)). It then falls to 72.0% and 71.4% in rounds 6 and 7. All three DiagEvo runs in this setting also attain their highest validation accuracy at round 5. Pure free exploration peaks at 69.5% in round 3, matching its ablation score (Table 2), then falls to 69.0%, 68.6%, and 68.0% in rounds 5, 6, and 7.

New error causes enter each round, raising the total from 151 to 244 (Figure 3(b)). End-of-round consolidation merges redundant skill nodes, whose count stabilizes near 36 after round 3. *Active* causes peak at 188 in round 2 and fall to 117 by round 5. By then, *Mastered* causes reach 127, over half of the memory. Promotion resets the closed episode's frequency and removes its past failures from F_t. The schedule then uses the updated F_t to balance cause-targeted generation with free exploration (Equation 1).

Pseudo-Label Quality. Figure 3(c) compares the three solvers after each round's training on 200 shared challenger-generated questions. Each solver forms a pseudo-label by majority vote over 12 responses per question, matching the response count for curriculum construction. Qwen3.8-Max proposes reference answers, which humans verify and correct. Pseudo-label accuracy is the percentage of pseudo-labels matching these human-verified oracle answers.

With pure free exploration and only the absolute confidence constraint, pseudo-label accuracy falls from 83% in round 1 to 65% in round 5. On the same round-5 questions, training with memory under the same constraint raises accuracy to 72%. Adding the relative confidence constraint during training raises it to 75%. These gains of 7 and 3 points support combining memory and double-confidence filtering during training to improve pseudo-label accuracy on shared questions. Appendix H details the evaluation protocols and analyzes question coverage, length, lexical diversity, and difficulty across rounds.

3.5 Case Study

Figure 4 traces a case on Qwen3-8B-Base. Two solver trajectories for $\log_2(x-1)+\log_2(x-5)=3$ derive $x = 3 \pm 2\sqrt{3}$. Only the trajectory agreeing with the pseudo-label checks $x > 5$. The diagnostician identifies the error cause as missing domain checks after algebraic transformation. It stores this cause as *Active* under the skill node *Checking Conditions after Algebraic Steps*.

In the next round, cross-state stitching pairs this cause with a *Mastered* cause from the same skill node: applying AM-GM without checking whether equality is possible. The resulting question asks for $\inf F + N$, where $F(x) = t + 1/t$ and N counts the inputs attaining the infimum. Here $t = \sqrt{(x-2)/(x+1)}$ with $x > 2$, so $0 < t < 1$ rules out AM-GM equality at $t = 1$. This tests the

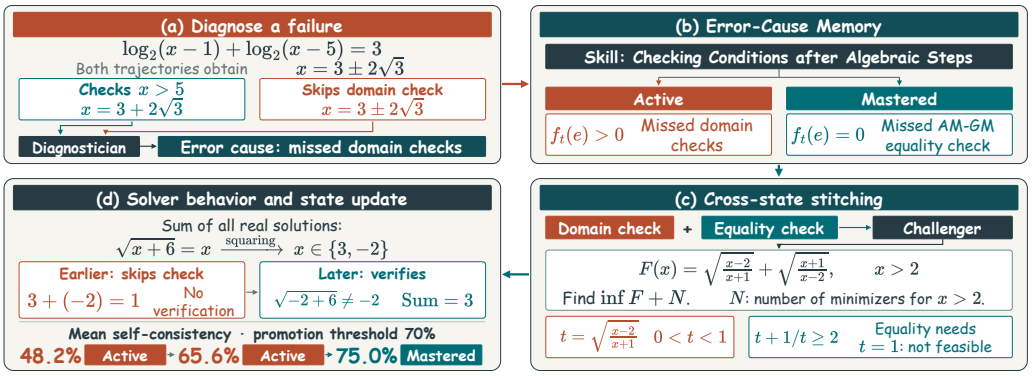


Figure 4: Case study on Qwen3-8B-Base. (a, b) Diagnosis records missing domain checks as an *Active* cause. (c) Cross-state stitching guides question generation. (d) Later responses verify candidates, and higher self-consistency promotes the cause to *Mastered*.

original checking rule in a new mathematical form. The question passed double-confidence filtering and entered solver training in round 4.

A later targeted question asks for the sum of solutions to $\sqrt{x+6}=x$. An earlier response accepts both 3 and -2, returning 1. A later response checks the original equation, rejects -2, and returns 3. Mean self-consistency on questions targeting this cause rises from 48.2% to 65.6% and then 75.0%, exceeding the 70% promotion threshold. The cause becomes *Mastered*, ending direct targeting while remaining available for cross-state stitching. The trace illustrates how self-play failures guide what to practice and how memory updates adjust later targeting. Appendix B provides the full trace.

4 Related Work

Self-Play. Self-play trains a challenger and solver on generated tasks (Huang et al., 2026; Zhao et al., 2025; Xia et al., 2026b). Some methods guide generation with human examples or documents (Yu et al., 2025; Liu et al., 2025a; Fan et al., 2026). SESA retrieves skills for its solver, while SPADE uses past environments to guide generation (Fu et al., 2026; Liu et al., 2026).

Memory and Curriculum Learning. Memory supports task solving (Shinn et al., 2023; Ouyang et al., 2026) and policy training (Xia et al., 2026a). TRACE targets weaknesses found on existing tasks (Kang et al., 2026). DiagEvo tracks recurring error causes as *Active* or *Mastered*. Their states and recurrence counts guide question generation without external task resources.

Label-Free Reinforcement Learning. Label-free RL uses answer agreement (Huang et al., 2026), reasoning consistency (Zhang et al., 2025a), or model confidence (Li et al., 2025) as reward signals. EMPO filters questions by semantic entropy (Zhang et al., 2025b). Co-rewarding uses question variants or a moving-average teacher, while Self-Harmony combines votes across question variants (Zhang et al., 2026b; Wang et al., 2026). DiagEvo's double-confidence filtering retains intermediate-difficulty questions only when the most common answer has a clear vote lead over the second. Appendix A provides additional comparisons.

5 Conclusion

DiagEvo turns recurring solver failures into guidance for self-play without external task resources. Its error-cause memory uses cause states and frequencies to balance cause-targeted generation with free exploration. Double-confidence filtering selects training questions. With the default 4B diagnostician, DiagEvo ranks first overall among the compared methods on all three solvers. Qwen3-8B reaches a 72.3% mathematical reasoning average, 4.5 points above R-Zero. Ablations support the contributions of memory guidance, adaptive scheduling, and filtering.

AI USE STATEMENT

We used generative AI tools to revise the manuscript, search and summarize related work, create figures, and modify code. We also used them to organize and summarize experimental data and obtain feedback on experiment design and result interpretation. DiagEvo uses language models to generate training questions and pseudo-labels and extract error causes (Section 2). We developed the method and derivations. We reviewed all AI-assisted content and take responsibility for the final text, code, figures, and results.

ETHICS STATEMENT

We study language model reasoning using self-generated mathematical training questions and existing benchmarks. We manually checked and corrected the reference answers used to evaluate pseudo-label accuracy.

REPRODUCIBILITY STATEMENT

Section 2 describes the method. Appendices D, E, and I provide the challenger reward, training configuration, and prompt templates. Appendix F specifies validation, checkpoint selection, benchmark evaluation, and baseline score validation. Section 3.3 and Appendices G and H report ablations, parameter sensitivity, and diagnostic evaluation protocols.

REFERENCES

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, Jeongyeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.30. URL https://aclanthology.org/2026.eacl-long.30/.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48. ACM, 2009. doi: 10.1145/1553374.1553380. URL https://icml.cc/2009/papers/119.pdf.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.829. URL https://aclanthology.org/2026.findings-acl.829/.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. AI4AI-Bench: Benchmarking LLM agents in algorithmic design for recursive self-improvement, 2026. URL https://arxiv.org/abs/2608.20318.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Xeron Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shuyue Guo, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, Sirun Li, Yizhi Li, Yunwen Li, Dehua Ma, Yuansheng Ni, Haoran Que, Qiyao Wang, Zhoufutu Wen, Siwei Wu, Tianshun Xing, Ming Xu, Zhenzhu Yang, Noah Wang, Junting Zhou, Yuelin Bai, Xingyuan Bu, Chenglin Cai, Liang Chen, Yifan Chen, Chengtuo Cheng, Tianhao Cheng, Keyi Ding, Siming Huang, Yun Huang, Yaoru Li, Yizhe Li, Zhaoqun Li, Tianhao Liang, Chengdong Lin, Hongquan Lin, Yinghao Ma,

Zhongyuan Peng, Zifan Peng, Qige Qi, Shi Qiu, Xingwei Qu, Shanghaoran Quan, Yizhou Tan, Zili Wang, Chenqing Wang, Hao Wang, Yiya Wang, Yubo Wang, Jiajun Xu, Kexin Yang, Ruibin Yuan, Yuanhao Yue, Tianyang Zhan, Chun Zhang, Jinyang Zhang, Xiyue Zhang, Owen Zhang, Yue Zhang, Yongchi Zhao, Xiangyu Zheng, Chenghua Zhong, Yang Gao, Zhoujun Li, Dayiheng Liu, Qian Liu, Tianyu Liu, Shiwen Ni, Junran Peng, Yujia Qin, Wenbo Su, Guoyin Wang, Shi Wang, Jian Yang, Min Yang, Meng Cao, Xiang Yue, Zhaoxiang Zhang, Wangchunshu Zhou, Jiaheng Liu, Qunshu Lin, Wenhao Huang, and Ge Zhang. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. In *Advances in Neural Information Processing Systems*, volume 38, pp. 125016–125089, 2025. doi: 10.52202/085713-3766. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/a3c5af1f56fc73eef1ba0f442739f5ca-Abstract-Datasets_and_Benchmarks_Track.html.

Shengda Fan, Xuyan Ye, and Yankai Lin. DARC: Decoupled asymmetric reasoning curriculum for LLM evolution, 2026. URL https://arxiv.org/abs/2601.13761.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3811. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Zenghuang Fu, Zhaoyang Li, Qiuyuan Ai, Haoyu Wu, Minghui Wu, Chenxu Zhao, Ante Wang, Guannan He, and Changwei Wang. Self-Play Meets Skill Evolution: Self-Evolving Search Agents that Pose, Solve, and Remember, 2026. URL https://arxiv.org/abs/2607.29468.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=96apU6YzSO.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Hangoo Kang, Tarun Suresh, Jon Saad-Falcon, and Azalia Mirhoseini. TRACE: Capability-Targeted Agentic Training, 2026. URL https://arxiv.org/abs/2604.05336.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. BIG-Bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, Vienna,

Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1285. URL https://aclanthology.org/2025.acl-long.1285/.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857, 2022. doi: 10.52202/068431-0278. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract.html.

Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan Oseledets. Confidence is all you need: Few-shot RL fine-tuning of language models, 2025. URL https://arxiv.org/abs/2506.06395.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond Pass@1: Self-play with variational problem synthesis sustains RLVR. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Wjf3OMJxpn.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning, 2025a. URL https://arxiv.org/abs/2510.24684.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, Mickel Liu, Hanfei Yu, Zhaorun Chen, Weiyan Shi, Paul Pu Liang, Luke Zettlemoyer, Yejin Choi, and Natasha Jaques. SPADE: Self-Play in Adaptive Synthetic Executable Environments, 2026. URL https://arxiv.org/abs/2608.19197.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, Vienna, Austria, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.694. URL https://aclanthology.org/2025.acl-long.694/.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. RSIBench-Data: Benchmarking data-centric research for recursive self-improvement, 2026. URL https://arxiv.org/abs/2607.25886.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

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. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025. doi: 10.48550/arXiv.2505.21444. URL https://arxiv.org/abs/2505.21444.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models, 2024. URL https://arxiv.org/abs/2404.14387.

Ru Wang, Wei Huang, Qi Cao, Yusuke Iwasawa, Yutaka Matsuo, and Jiaxian Guo. Self-Harmony: Learning to harmonize self-supervision and self-play in test-time reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/86178c06fd59ebc58dc31f8edb3c54af-Abstract-Conference.html.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. EvolveR: Self-evolving LLM agents through an experience-driven lifecycle, 2026. URL https://arxiv.org/abs/2510.16079. Accepted by ICML 2026.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning, 2026a. URL https://arxiv.org/abs/2602.08234.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing Self-Evolving Agents from Zero Data via Tool-Integrated Reasoning. In *Third Conference on Language Modeling*, 2026b. URL https://arxiv.org/abs/2511.16043.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving LLMs with minimal human supervision, 2025. URL https://arxiv.org/abs/2512.02472.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 59849–59887, 2025a. doi: 10.52202/085713-2002. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/565f995643da6329cec701f26f8579f5-Abstract-Conference.html.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised LLM reasoning incentivization. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-2259. URL https://papers.nips.cc/paper_files/paper/2025/hash/614f8eba720cfc7ff00274bd64fb0a3f-Abstract-Conference.html.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8587c39571f3e327725e2812749a9389-Abstract-Conference.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3534. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9837dc00ff67d176373268ed48042d49-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4376. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/be690ea16f005c174f6c4102a5970e67-Abstract-Conference.html.

zwhe99. AMC23. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/zwhe99/amc23. Accessed 2026-08-29.

A Extended Related Work

A.1 Self-Play

Self-play couples task generation with solver updates. R-Zero generates mathematical questions and trains its challenger with solver uncertainty and a repetition penalty (Huang et al., 2026). Absolute Zero generates program-based tasks and uses code execution to assess learnability and solution correctness (Zhao et al., 2025). Agent0 adds tool use to this loop. Its curriculum reward combines solver uncertainty, tool calls, and repetition penalties. The executor trains with majority-vote pseudo-labels (Xia et al., 2026b).

Other methods use external task resources to guide generation. R-Few uses labeled examples both to guide generation and to train the solver (Yu et al., 2025). SPICE generates questions and reference answers from documents (Liu et al., 2025a). DARC trains its questioner with explicit difficulty targets, then builds an offline curriculum. Its teacher shares the student's parameters but also sees the source documents (Fan et al., 2026).

Recent methods also retain experience across self-play rounds. SESA distills failed search trajectories into skills retrieved by the solver. This changes solver success rates and thereby the challenger's generation rewards (Fu et al., 2026). SPADE stores past executable environments, skill tags, and regret scores to guide environment generation. Its regret signal measures the return gap with and without a hint, while external corpora supply task content (Liu et al., 2026). DiagEvo stores recurring error causes and directly conditions new questions on them. Cause states and recurrence counts determine which weaknesses to revisit and when to favor free exploration.

A.2 Memory and Curriculum Learning

Experience memory can guide later decisions without changing model weights. Reflexion records feedback for later attempts, while ExpeL compares successful and failed trajectories to revise a shared list of insights (Shinn et al., 2023; Zhao et al., 2024). AutoGuide compares trajectories with different rewards and extracts instructions for their shared context before the actions diverge (Fu et al., 2024). CER distills environment dynamics and decision skills, then retrieves and updates this memory during interaction (Liu et al., 2025b). ReasoningBank extracts strategies from both successful and failed trajectories (Ouyang et al., 2026). ReMe refines procedural memory using retrieval frequency and downstream success (Cao et al., 2026). ACE maintains a context playbook through reflection and local updates (Zhang et al., 2026a).

Memory can also evolve alongside model parameters. SkillRL couples a skill library with policy training and revises skills in response to validation failures (Xia et al., 2026a). EvolveR alternates experience distillation with policy training. Retrieved principles guide actions, while use and success counts guide memory updates (Wu et al., 2026). These methods help the policy use past experience while solving tasks. DiagEvo uses error-cause memory to guide the challenger's question generation.

Curriculum learning controls which examples a model encounters and in what order (Bengio et al., 2009). Online difficulty filtering selects intermediate-difficulty questions from an existing pool using the current policy's success rate (Bae et al., 2026). SvS synthesizes variants of labeled questions from correct solutions while preserving their answers (Liang et al., 2026). TRACE compares successful and failed trajectories in target environments to identify missing capabilities. It generates verifiable training environments for these capabilities and combines separately trained adapters (Kang et al., 2026). DiagEvo extracts error causes from self-play trajectories and tracks them as *Active* or *Mastered* across rounds. These states guide question generation without external task resources.

A.3 Label-Free Reinforcement Learning

Label-free RL derives supervision from model outputs. R-Zero uses majority-vote pseudo-labels for generated questions (Huang et al., 2026), while TTRL applies the same reward principle to adaptation on supplied test questions (Zuo et al., 2025). CoVo combines consistency and volatility in how intermediate reasoning states support candidate answers (Zhang et al., 2025a). RLSC instead weights responses by model probabilities, without extracting majority-vote labels (Li et al., 2025).

Several methods improve the reliability of these internal signals. EMPO groups answers by meaning and rewards responses according to their group's frequency. It filters questions using lower and upper semantic-entropy thresholds (Zhang et al., 2025b). SRT updates majority-vote pseudo-labels during training and reports gains followed by collapse in longer runs. Its consistency-based filtering delays collapse in the tested setting (Shafayat et al., 2025). Co-rewarding uses agreement across question variants, a moving-average teacher, or both to stabilize supervision (Zhang et al., 2026b). Self-Harmony selects pseudo-labels using the harmonic mean of answer frequencies for original questions and their variants (Wang et al., 2026). DiagEvo keeps majority-vote supervision. It filters questions by difficulty and the vote ratio between the two most common answers. These checks address vote ambiguity in questions generated by the evolving curriculum.

B Case Study: From Failure to Curriculum

This appendix traces one error cause across rounds (Figure 4), following it through diagnosis, cause-targeted generation, and the subsequent state transition.

Round t: failure and diagnosis. For $\log_2(x-1)+\log_2(x-5)=3$, both trajectories derive $x=3\pm2\sqrt{3}$, but only the trajectory agreeing with the pseudo-label enforces $x>5$. The diagnostician records the transferable cause *"Fails to check whether candidate solutions remain valid under the original domain restrictions after algebraic transformation."* It enters memory as *Active* under the skill node *Checking Conditions after Algebraic Steps*, alongside the *Mastered* cause *"Applies AM-GM to derive a lower bound but does not check whether an allowed input satisfies the equality condition."*

Round $t+1$: cause-targeted generation and training. The challenger samples this *Active* cause and combines it with the related *Mastered* sibling from the same skill node to generate

$$F(x)=\sqrt{\frac{x-2}{x+1}}+\sqrt{\frac{x+1}{x-2}}, \qquad x>2.$$

Let N be the number of real $x>2$ at which F attains its infimum. Find $\inf F+N$. The question requires propagating $0<t<1$ for $t=\sqrt{(x-2)/(x+1)}$ and recognizing that equality at $t=1$ is impossible. It tests the same corrective action under a new surface form. The question passed double-confidence filtering and entered solver training in round 4.

Later rounds: behavior and state. A later targeted question asks for the sum of all real solutions to $\sqrt{x+6}=x$. Squaring gives $(x-3)(x+2)=0$. A solver response from an earlier round returns $3+(-2)=1$ without checking the candidates; a response from a later round substitutes them into the original equation, rejects -2 because $\sqrt{-2+6}=2\neq-2$, and returns 3. Across rounds, mean self-consistency on questions targeting this cause rises from 48.2% to 65.6% and then 75.0%. The final value exceeds $\theta_{\text{up}}=70\%$, promoting the cause to *Mastered*. The trace illustrates how a diagnosed failure changes the curriculum and later solver behavior.

C Limitations and Future Work

Pseudo-label confidence. Double-confidence filtering retains questions whose majority vote share lies within the chosen range and whose leading answer is clearly ahead of the second answer. Both conditions measure agreement among solver responses rather than ground-truth correctness. A shared error can therefore receive strong agreement and pass the filter. Future work can study other verification signals produced during self-play to identify such cases.

Training horizon. DiagEvo currently sets the number of co-evolution rounds in advance, while its memory controls question selection within each round. In the seven-round analysis, the mathematical average is highest at round 5. Future work can use changes in the set of *Active* causes and solver self-consistency to decide whether another round is useful.

Evaluation scope. DiagEvo builds its curriculum from mathematical questions. The gains on general-reasoning benchmarks show transfer from mathematical training, but do not test direct curriculum construction in other domains. Future work can extend the diagnosis and memory loop to open-ended tasks with long interaction sequences, including multi-turn tool use (Jin et al., 2025).

D Challenger Update and Quality Reward

Following R-Zero (Huang et al., 2026), each round uses one batch to update the challenger and another to build the solver training set. The current challenger θ_t first samples $\mathcal{Q}^{\text{chal}}_{t+1}\sim q_{t+1}(\cdot\mid\mathcal{M}_t;\theta_t)$. The frozen solver answers this challenger-update batch. Group Relative Policy Optimization (GRPO) then updates the challenger to θ_{t+1}. The updated challenger samples a new pool of curriculum candidates $\mathcal{Q}^{\text{cur}}_{t+1}\sim q_{t+1}(\cdot\mid\mathcal{M}_t;\theta_{t+1})$. Section 2.2 uses this pool to construct and filter the solver training set.

For a format-valid question x, let $p_1(x)$ denote the frozen solver's majority-answer vote share. The uncertainty reward is

$$r_{\text{unc}}(x) = 1 - 2\left|p_1(x) - \frac{1}{2}\right|. \tag{9}$$

It peaks at 50% self-consistency, favoring questions near the solver's competence boundary.

The repetition penalty in Equation 11 follows R-Zero (Huang et al., 2026). For the challenger-update batch $\mathcal{X} = \{x_i\}_{i=1}^{B}$, we compute the pairwise distance

$$d_{ij} = 1 - \text{BLEU}(x_i, x_j). \tag{10}$$

Agglomerative clustering with the R-Zero criterion produces clusters $\mathcal{C} = \{C_1, \ldots, C_K\}$. For $x_i \in C_k$, the repetition penalty is $r_{\text{rep}}(x_i) = |C_k|/B$, so large clusters receive larger penalties.

The full reward is

$$r_{\text{chal}}(x) = \max\{0,\ r_{\text{unc}}(x) - \lambda r_{\text{rep}}(x)\}, \tag{11}$$

with $\lambda = 1$ following R-Zero; format-invalid questions receive zero reward.

E Training Hyperparameters

This section reports the settings used to reproduce DiagEvo. We first list the GRPO configuration shared by the challenger and solver, then report the response-group and co-evolution settings, followed by the reward, filtering, and curriculum thresholds.

GRPO Optimization. Both the challenger and solver use the learning rate, weight decay, KL penalty coefficient, rollout temperature, and top-p reported by R-Zero, as summarized in Table 3. Their role-specific batch sizes, response-group sizes, and update settings are reported separately in Table 4. The surrogate objective uses the default clip range in the standard implementation. All experiments run on a single node of 8 NVIDIA H200 GPUs with BF16 mixed precision and FlashAttention 2.

Table 3: GRPO optimization settings for the challenger and solver.

Setting	Value
Learning rate	1×10^{-6}
Weight decay	1×10^{-2}
KL penalty coefficient β	1×10^{-2}
Rollout temperature	1.0
Rollout top-p	0.99

Batch Size and Response Groups. In each round, the challenger is first optimized for up to 6 GRPO steps. Each challenger step uses a global batch size of 256 and a group size of eight. The updated challenger then generates the curriculum candidates. For every candidate, the current solver samples 12 construction responses to determine the pseudo-label, p_1, and p_2. For each *Active* cause, the p_1 values of all candidates targeting that cause are averaged to obtain the self-consistency score used for promotion. Candidates whose construction votes satisfy both confidence constraints form the solver training set. For each retained question, one construction response agreeing with its pseudo-label is stored as the reference trajectory for diagnosis. The solver is then optimized for up to 12 GRPO steps. Each solver step uses a global batch size of 256. For each retained question in the batch, the solver samples a separate group of eight optimization responses. These responses provide the policy-gradient signal, while their failed trajectories are paired with the stored reference for diagnosis and memory maintenance. Table 4 summarizes these settings.

Table 4: DiagEvo-specific training configuration (challenger and solver co-evolution loop).

Hyperparameter	Value
Solver GRPO global batch size	256
Challenger GRPO global batch size	256
Solver GRPO group size G	8
Challenger GRPO group size	8
Solver max steps per round	12
Challenger max steps per round	6
Construction responses per candidate N	12
Main-experiment training rounds	5

Reward, filtering, curriculum, and memory-maintenance hyperparameters. Table 5 reports the numerical settings that control the challenger reward, double-confidence filtering, curriculum transitions, and memory maintenance. Appendix I specifies the diagnostician routing and retrieval procedure.

Table 5: Reward, filtering, curriculum, and memory-maintenance hyperparameters used throughout DiagEvo.

Symbol	Value
Repetition penalty weight λ	1.00
Absolute difficulty lower bound p_{low}	0.25
Absolute difficulty upper bound p_{high}	0.75
Relative confidence threshold τ	1.60
State promotion threshold θ_{up}	0.70
Exploration-exploitation proportionality k	0.50
Deduplication similarity threshold θ_{dup}	0.50
Node-merge similarity threshold θ_{merge}	0.50
Node candidate shortlist size K	5

Implementation Details. All three Qwen3-Instruct-2507 diagnosticians run in non-thinking mode. Qwen3-4B-Instruct-2507 is the default for experiments that do not compare diagnostician scale. For questions retained by double-confidence filtering, the diagnostician compares failed optimization trajectories with stored construction references. It reads only questions, pseudo-labels, and trajectories produced during self-play. The embedding model reads only error causes derived from these trajectories. It uses text-embedding-v4 for error-cause retrieval. Both models are frozen and general-purpose. Section 2.3 describes how these models are used for extraction and deduplication.

Training Time Breakdown. Table 6 reports each component's share of total elapsed time in a complete DiagEvo training run. Question sampling, pseudo-label construction, and the GRPO updates of the two models account for 94.2% of the total time; each of these components has a direct counterpart in R-Zero. Diagnosis and memory maintenance account for 5.7% of total training time, with 3.9% spent on comparative error-cause extraction.

Table 6: Training time breakdown for a complete DiagEvo run with the default configuration. Diagnosis and memory maintenance, the components specific to DiagEvo, account for 5.7% of the total time.

Component	Training time (%)
Challenger policy optimization	27.1
Curriculum candidate generation	10.9
Pseudo-label construction and double-confidence filtering	30.3
Solver policy optimization	25.9
Diagnosis and memory maintenance	**5.7**
Comparative error-cause extraction	3.9
Deduplication and node assignment	1.3
Skill-node consolidation	0.4
State–frequency update and scheduling	0.1
Other orchestration	0.1

F Evaluation Protocol and Baseline Score Validation

Validation and Checkpoint Selection. The fixed validation set contains 600 questions. We sample 300 Level 3 and Level 4 questions from the MATH training split (Hendrycks et al., 2021b), covering all seven subject categories. Another 300 questions come from the MMLU validation split (Hendrycks et al., 2021a), covering all 57 subjects. These questions are disjoint from the nine test benchmarks. We tune hyperparameters using accuracy over all 600 validation questions and fix the defaults before component ablations. Each run uses the checkpoint with the highest validation accuracy for evaluation on all nine test benchmarks.

Benchmark Evaluation. For mathematical reasoning, we follow the simple-evals protocol and use GPT-4o as an automatic judge, matching DARC (Fan et al., 2026). AMC contains 40 questions. Following R-Zero (Huang et al., 2026), we sample 32 responses per AMC question and report their mean accuracy (mean@32). For general reasoning, we use exact-match accuracy with greedy decoding.

Baseline Score Validation. Four baseline rows in Table 1 are taken from DARC (Fan et al., 2026). To verify that these scores are comparable to our own runs, we re-evaluate the three base models and reproduce R-Zero (Huang et al., 2026) under our own infrastructure. We use the same solvers, benchmarks, and evaluation protocol. Table 7 compares the aggregate scores. Base-model averages differ by at most 0.1 points, and R-Zero scores differ by at most 0.5 points. Per-benchmark deviations remain below 1.0 points. Table 1 therefore reports our own scores for the base models and R-Zero. It reports the scores from DARC for the remaining baselines. DARC does not report R-Few for OctoThinker-8B-Hybrid-Base, so Table 1 omits this setting.

G Generation Schedule and Hyperparameter Sensitivity

This appendix compares fixed and adaptive generation and examines sensitivity to k (Equation 1) and τ (Equation 3). All runs use Qwen3-8B-Base with the default 4B diagnostician.

Fixed and Adaptive Generation. We compare the frequency-driven schedule with fixed ratios of free exploration to cause-targeted generation, keeping all other settings unchanged (Table 8). The schedule achieves higher mathematical and general averages than all three fixed ratios.

Hyperparameter Sensitivity. The default values, $k = 0.5$ and $\tau = 1.6$, are selected on the validation set. Each sensitivity experiment varies one of them while keeping other settings fixed and reports test benchmark scores.

Table 9 varies k in the frequency-driven schedule. Smaller values favor earlier exploitation, whereas larger values preserve more exploration.

Table 7: Aggregate scores of the reported baselines and of our reproduction under the same evaluation protocol. Base-model averages match within 0.1 points and R-Zero scores within 0.5 points on every column.

Solver	Scores	Math Avg	Gen Avg	Overall
Qwen3-4B-Base	Base (reported)	53.1	27.9	41.9
	Base (reproduced)	53.0	27.8	41.8
	R-Zero (reported)	61.1	32.2	48.2
	R-Zero (reproduced)	60.6	31.9	47.9
Qwen3-8B-Base	Base (reported)	63.3	33.1	49.9
	Base (reproduced)	63.4	33.2	50.0
	R-Zero (reported)	67.6	36.3	53.7
	R-Zero (reproduced)	67.8	36.2	53.8
OctoThinker-8B-Hybrid-Base	Base (reported)	35.8	10.6	24.6
	Base (reproduced)	35.9	10.7	24.7
	R-Zero (reported)	46.4	21.2	35.2
	R-Zero (reproduced)	46.0	20.9	34.9

Table 8: Fixed and adaptive generation. Ratios denote free exploration to cause-targeted generation.

Generation policy	Math Avg.	General Avg.
Fixed 7:3	70.1	36.9
Fixed 5:5	70.9	37.2
Fixed 3:7	71.2	37.6
Frequency-driven (full)	**72.3**	**38.8**

Table 9: Sensitivity of the frequency-driven schedule to k (Qwen3-8B-Base).

k	Math Avg	General Avg
0.2	71.4	38.1
0.5 (default)	**72.3**	**38.8**
1.0	71.8	38.4
2.0	71.3	37.8

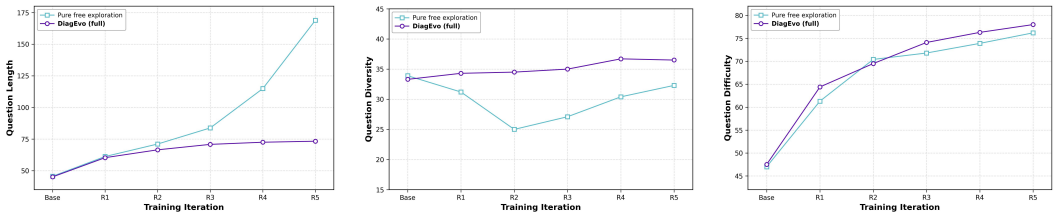
Table 10 varies τ in double-confidence filtering; the $\tau = 1.6$ row repeats the full configuration from Table 2 for reference. Performance falls on either side of $\tau = 1.6$. The drop under stricter filtering is consistent with removing too many training questions, although we do not directly measure retention rates.

Table 10: Sensitivity of double-confidence filtering to the relative confidence threshold τ (Qwen3-8B-Base).

τ	Math Avg	General Avg
1.2 (loose)	71.6	38.4
1.6 (default)	**72.3**	**38.8**
2.0 (strict)	71.3	38.1

H Diagnostic Evaluation Protocols

This section specifies the measurements behind Figures 3, 5, and 6. All of them use Qwen3-8B-Base with the default 4B diagnostician, and every variant shares the training configuration in Appendix E.



(a) Question length (b) 2-gram diversity (c) Question difficulty

Figure 5: Properties of questions generated after each round on Qwen3-8B-Base. (a) Average length in words. (b) Lexical diversity over adjacent word pairs, reported as 2-gram diversity. (c) Difficulty measured by the error rate of a fixed Qwen3-8B-Base solver against oracle answers. Both variants use double-confidence filtering. *Pure free exploration* removes the error-cause memory.

Per-round performance. After each round, we evaluate the current solver under the protocol in Appendix F and report the mathematical reasoning average. Both curves in Figure 3(a) retain double-confidence filtering and differ only in whether question generation uses the error-cause memory. We additionally evaluate both variants in rounds 6 and 7 under the same protocol; Section 3.4 reports the resulting averages.

Memory statistics. We count the *Active* and *Mastered* error causes and the skill nodes in the memory committed after each round (Equation 4).

Pseudo-label accuracy. We track pseudo-label accuracy across rounds, as in R-Zero (Huang et al., 2026). In each round, we select 200 challenger-generated questions and use the same set for all three variants. Each variant is evaluated using its solver checkpoint after that round's training. For every question, the solver samples 12 responses and takes their majority answer as the pseudo-label, matching the response count for curriculum construction. Qwen3.8-Max proposes each reference answer by majority vote over 16 sampled responses. Humans verify every answer and correct any errors to obtain the oracle answers, which are used only for evaluation. We report the percentage of each variant's majority-vote pseudo-labels that match these oracle answers across all 200 questions.

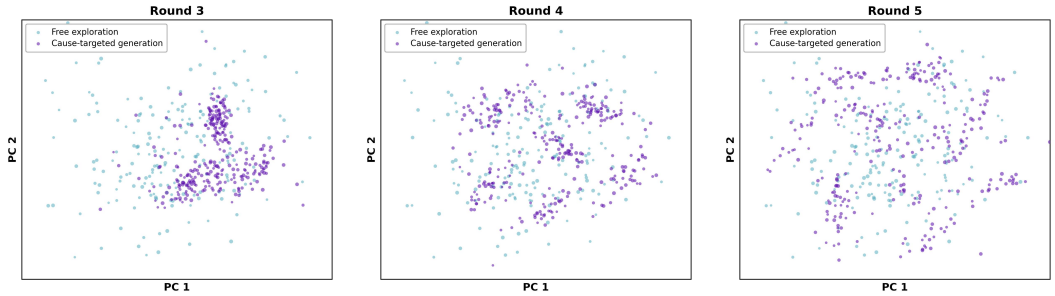
Question properties. Following the protocol of R-Few (Yu et al., 2025), we measure the average question length in words and the 2-gram lexical diversity of the questions generated after each round. Difficulty is the error rate of a fixed Qwen3-8B-Base solver on these questions, with answers relabeled by Qwen3.8-Max; since this oracle differs from the one used by R-Few, the absolute values should not be compared across papers. As in Figure 3(a), both variants retain double-confidence filtering.

Figure 5 tests whether question difficulty grows with lexical diversity or simply with question length. Without the memory, average length grows from 45.6 to 168.9 words. The 2-gram diversity rises again only after the questions become much longer. This timing links the late increase in 2-gram diversity to the added length.

With the memory, average length stays near 73 words after round 2. At the same time, 2-gram diversity rises from 33.3 to 36.5. Both variants generate harder questions over the rounds. The memory-guided variant does so at a stable length. Its increase in difficulty therefore cannot be explained by longer questions.

Coverage visualization. We embed the questions generated after each round with text-embedding-v4 and project the embeddings to two dimensions with PCA. Figure 6 colors each question by its generation mode, free exploration or cause-targeted.

The changing memory also changes what the challenger generates (Figure 6). Free exploration covers a broad region that remains stable across rounds. Cause-targeted generation moves into new regions as the memory gains error causes. The two modes therefore have complementary roles. Free exploration maintains broad coverage, while cause-targeted generation follows the solver's current error causes. This pattern is consistent with the performance loss when Table 2 removes either mode.



(a) Question coverage, round 3 (b) Question coverage, round 4 (c) Question coverage, round 5

Figure 6: Principal component analysis (PCA) projections of questions generated in rounds 3–5. Colors indicate the generation mode.

I Prompt Templates and Memory Routing

This section specifies the runtime prompts and routing procedures for the challenger, solver, and diagnostician. Challenger and solver prompts follow the level of detail in R-Zero (Huang et al., 2026); diagnostician prompts implement the semantic decisions described in Section 2.3. Fields marked `<xxx>` are replaced at runtime.

Design rationale. Pseudo-labels come from solver majority voting (Section 2.2), so the challenger does not provide answers. We retain the role, difficulty criterion, and `<question>` format of R-Zero (Huang et al., 2026), but omit its `\boxed{final answer}` requirement because DiagEvo constructs pseudo-labels from solver votes. Diagnostic context appears only in the challenger user message: either one error cause or an *Active*/*Mastered* pair. The shared system message is unchanged across generation modes. Each cause is accompanied by its parent skill-node label to provide broader knowledge context.

Shared System Message. The following system message is used for all three generation modes described below.

> You are an expert competition-math problem setter. FIRST, in your private scratch-pad, think step-by-step to design a brand-new, non-trivial problem. The problem could come from any field of mathematics, including but not limited to algebra, geometry, number theory, combinatorics, prealgebra, probability, statistics, and calculus. Aim for a difficulty such that fewer than 30% of advanced high-school students could solve it. Avoid re-using textbook clichés or famous contest problems.
>
> THEN, without revealing any of your private thoughts, output *exactly* the following block:
>
> `<question>`
>
> {The full problem statement on one or more lines}
>
> `</question>`
>
> Do NOT output anything else—no explanations, no extra markup.

Mode 1: Free Exploration. The challenger relies solely on its own internal knowledge, with no reference to the memory (Section 2.1).

> **User Message:** Generate one new, challenging reasoning question now. Remember to format the output exactly as instructed.

Mode 2: Cause-Targeted Generation (Single Cause). Activated when the sampled *Active* error cause's skill node contains no *Mastered* causes (Section 2.1).

> **User Message:** Generate one new, challenging reasoning question now. The question must specifically target the following reasoning weakness observed in the solver's failures:
>
> Knowledge area: `<skill_node>`
>
> Weakness to probe: `<error_cause>`
>
> Design the question so that correctly solving it requires the solver to overcome this specific weakness. Remember to format the output exactly as instructed.

Mode 3: Cross-State Stitching. Activated when the sampled *Active* error cause's skill node also contains at least one *Mastered* cause (Section 2.1); the two causes are combined in a single composite question rather than presented as separate tasks.

> **User Message:** Generate one new, challenging reasoning question now. The question must weave together two reasoning elements within the same knowledge area into a single, coherent problem, rather than testing them separately:
>
> Knowledge area: `<skill_node>`
>
> Element A (an observed weakness that remains *Active* and should still be challenging): `<active_cause>`
>
> Element B (a previously diagnosed weakness for which the solver has reached high self-consistency, as measured by the endogenous consistency proxy; the question should require the corresponding corrective reasoning skill): `<mastered_cause>`
>
> Design a single self-contained question that genuinely requires combining both elements to reach the answer—neither element should be solvable in isolation from the other. Remember to format the output exactly as instructed.

Solver Prompt. The solver prompt is identical to that of R-Zero (Huang et al., 2026) because answer extraction is independent of question generation. R-Few (Yu et al., 2025) uses the same instruction, while SPICE (Liu et al., 2025a) instead uses model-specific reasoner templates.

> **System Message:** Please reason step by step, and put your final answer within `\boxed{}`.
>
> **User Message:** `<question>`

Diagnostician Prompts. The diagnostician operates within a hybrid code-and-LLM pipeline that follows the memory-maintenance sequence in Section 2.3. Code performs similarity screening, top-K retrieval, and routing; the diagnostician performs extraction, deduplication, node assignment, and consolidation. Recurrence verification is folded into extraction. The diagnostician returns JSON outputs in the runtime order described below. Error causes and node labels must be 10–20-word verb–object phrases without question-specific values or variables, which keeps their embeddings focused on transferable semantics.

Pre-Check: Known-Cause Recurrence Verification (LLM Call #1, folded into Prompt 1). For each question retained by Equation 3, diagnosis pairs failed optimization trajectories with the stored construction reference that agrees with its pseudo-label. No extra responses are sampled for diagnosis. The known-cause list is empty for free exploration, contains the sampled *Active* cause for single-cause generation, and contains both the sampled *Active* cause and the stitched *Mastered* cause for cross-state generation. Prompt 1 first locates the earliest reasoning difference and compares it with every supplied cause. It assigns each failed trajectory to at most one supplied cause, selecting the single closest match when more than one appears plausible. A confirmed match contributes one count to $\Delta_t(e)$ for the matched cause; otherwise, the prompt extracts a new candidate cause for memory matching. Both paths use the same LLM call.

Prompt 1: Error-Cause Extraction (LLM Call #1).

> **System Message:** You are a diagnostician analyzing why a math solver's reasoning went wrong. You will be shown a problem, a working reference answer, and two

solving attempts that disagree with each other: one that reaches the reference answer and one that does not. Your job is to locate the exact reasoning step where the two attempts diverge and name the underlying skill deficiency responsible for the divergence.

Do not re-solve the problem yourself, and do not treat the reference answer as ground truth: it was itself produced by majority voting over the solver's own samples and can occasionally be wrong. Your only task is to explain the divergence between the two given attempts, not to adjudicate correctness from outside evidence.

If one or more known causes are supplied below, compare the located divergence with every supplied cause. Select at most one cause whose underlying failure mechanism best explains the earliest divergence. Only if none of the supplied causes matches should you characterize a new one.

EXTRACTION PRINCIPLES:

- Localize the single earliest step at which the two attempts stop agreeing, and diagnose that step specifically—do not summarize the disagreeing attempt's overall approach.
- State the error cause as a general, transferable skill deficiency, phrased as a single verb–object phrase (e.g., "misapplies triangle inequality to algebraic side lengths"), never as a restatement of this problem's specific numbers, entities, or context.
- The description must be 10–20 words. This is a hard constraint, not a suggestion: it will later be embedded and used for cosine-similarity screening before semantic confirmation, so any leftover problem-specific detail, hedge, or explanation will dilute the embedding and silently corrupt every downstream deduplication and routing decision.
- If the two attempts do not actually diverge in reasoning strategy (e.g., the disagreement is a trivial arithmetic or copying slip), report this explicitly rather than forcing an artificial skill label onto a non-skill failure.

User Message: # Problem `<problem>`

Working Reference Answer `<pseudo_label>`

Attempt That Agrees With the Reference `<agreeing_response>`

Attempt That Disagrees With the Reference `<disagreeing_response>`

Known Causes Used for Generation (omit this section if the problem was freely generated) `<known_cause_list>`

Each entry contains a cause ID, its current state, and its cause description.

OUTPUT FORMAT: Return exactly one JSON object with the following fields.

```
{
  "matched_cause_id": null,
  "error_cause": null
}
```

Set `matched_cause_id` to the exact ID of the single supplied cause that best explains the earliest divergence, or to `null` if none matches. When `matched_cause_id` is non-null, set `error_cause` to `null`; the pipeline assigns the observation directly to that cause and skips the stages below. When `matched_cause_id` is `null`, `error_cause` must be a new 10–20-word verb–object description unless the failure is unrelated to any identifiable skill, in which case it is also `null`.

Stage 2a: Error-Cause Similarity Screening (non-LLM). For each new candidate cause, we compare its embedding with every stored cause. Candidates above θ_{dup} proceed to Prompt 2 for semantic confirmation. If none qualifies, the candidate proceeds directly to node assignment.

Prompt 2: Deduplication (LLM Call #2). Invoked only when Stage 2a returns at least one candidate.

System Message: You are a diagnostician deciding whether a newly identified error cause describes a mistake already recorded in the memory. You will be shown one new error cause and a short list of existing candidates that an embedding search flagged as textually similar to it. Judge duplication by underlying mechanism, not by wording: two causes represent the same error cause if fixing one would also fix the other, even when phrased differently; they are different if a solver could overcome one while still exhibiting the other.

Your decision directly controls the memory's current-episode frequency signal and may reactivate a matched *Mastered* cause. Frequency determines sampling weights within the Active cause-targeted pool and contributes to the next round's exploration–exploitation ratio: confirming a genuine duplicate adds evidence to an *Active* cause or starts a reactivated cause at one, while wrongly accepting two distinct failure modes as one would blur the signal used for cause-targeted generation. Err toward keeping mechanistically distinct causes apart whenever the match is not clearly correct.

User Message: # New Error Cause `<new_cause>`

Candidate Existing Causes `<candidate_cause_list>`

OUTPUT FORMAT: Return exactly one of the following valid JSON objects.

```
{
  "duplicate": true,
  "matched_cause_id": "<matched_cause_id>"
}
```

or

```
{
  "duplicate": false,
  "matched_cause_id": null
}
```

If more than one candidate plausibly matches, select the single closest one. When `duplicate` is `true`, the observation is assigned to `matched_cause_id` and contributes one count to $\Delta_t(e)$ in Equation 8; no new cause is created. The joint transition then increments an *Active* cause or starts a new active episode for a *Mastered* cause with the matched-failure count as its frequency. When `duplicate` is `false`, the new cause proceeds to Stage 3a for node assignment.

Stage 3a: Node Candidate Retrieval (non-LLM). We reuse the cause-level similarities from Stage 2a to retrieve candidate skill nodes. A node receives the maximum similarity between the new cause and any of its members, and the top-K nodes form the shortlist. If the memory has at most K nodes, all nodes are returned.

Prompt 3: Skill Node Assignment (LLM Call #3). Prompt 3 is invoked once for each new error cause that clears deduplication, using the top-K skill-node candidates from Stage 3a.

System Message: You are a diagnostician organizing error causes under skill nodes. Each node names a reasoning or knowledge skill that the challenger can target on its own. You will see a new error cause and its top-K most relevant existing skill nodes. Each node is shown with its label and one representative error cause. Base your decision only on this shortlist; do not assume access to the full memory. In a single pass, decide whether the new cause belongs to a listed node or needs a new one. Use the following three criteria:

- **Semantic distinctness:** is the new cause's underlying skill clearly different from every candidate node's skill, as defined relative to this specific candidate set (this criterion is trivially satisfied when no candidates are shown)?
- **Independent actionability:** would practicing to fix this cause be a meaningfully different exercise from practicing any candidate node's skill? A node should correspond to one coherent thing the solver can get better at, not a loose bag of unrelated causes.

- **Generality and extensibility:** if a new node is created, is its label general enough to later host other, distinct error causes that share this skill, rather than merely restating this one instance? A node opened too narrowly will never accumulate enough error causes to reveal a stable pattern for later composite-problem generation.

User Message: # New Error Cause `<new_cause>`
Top-K Candidate Skill Nodes `<candidate_node_list>`
OUTPUT FORMAT: Return exactly one of the following valid JSON objects.

```
{
  "create_new_node": true,
  "new_node_label": "<10-20 word verb-object label>",
  "assigned_node": null
}
```

or

```
{
  "create_new_node": false,
  "new_node_label": null,
  "assigned_node": "<assigned_node_id>"
}
```

Exactly one of `new_node_label` and `assigned_node` must be non-null. This single pass is the only chance to get the decision right: it is trusted and executed as-is, with no second retrieval or second diagnostician call to verify it, so weigh all three criteria within this one judgment rather than treating creation and assignment as separate questions.

Stage 3.5: Routing Execution (code only, no LLM or embedding call). Code executes Prompt 3's routing decision directly. If `create_new_node` is `true`, it creates a skill node labeled `new_node_label` and attaches the error cause as its first member; otherwise, it attaches the error cause under `assigned_node`. Either way, the new error cause is initialized with $(s_t(e), f_t(e)) = (\textit{Active}, 1)$, consistent with Equation 4.

Stage 4a: Node-Pair Candidate Screening (non-LLM). After all failures in round t are processed and routed, we compare skill-node embeddings pairwise. Pairs above θ_{merge} proceed to Prompt 4 for semantic confirmation.

Prompt 4: End-of-Round Consolidation (LLM Call #4). Prompt 4 is invoked once for each candidate pair from Stage 4a.

System Message: You are a diagnostician checking skill nodes for redundancy as the memory grows. Prompt 3 assigns each error cause using only the shortlist available at that time. Nodes created at different times may therefore describe the same skill. Your task is to identify these duplicate nodes. You will see two skill nodes, each with its label and all its error causes. Decide whether they describe the same skill and should be merged, or distinct skills that share similar words. Base your decision on all error causes under both nodes. Similar labels can cover different mistakes, while different labels can cover the same mistake.

User Message: # Skill Node A `<node_a_label>`: `<node_a_causes>`
Skill Node B `<node_b_label>`: `<node_b_causes>`
OUTPUT FORMAT: Return exactly one of the following valid JSON objects.

```
{
  "merge": true,
  "merged_label": "<10-20 word verb-object label>"
}
```

or

```
{
  "merge": false,
  "merged_label": null
}
```

The merged label must remain a general, transferable skill description under the same 10–20 word verb–object phrase constraint as any node label, not a concatenation of A's and B's labels. When `merge` is `true`, the pipeline relabels the surviving node as `merged_label` and re-parents every error cause from both A and B under it without an additional cause-level deduplication pass; cross-node duplicate screening has already been applied in Stage 2a. Individual cause states and current-episode frequencies remain unchanged.