LEARNING FROM EVOLVING ERRORS: ADAPTIVE ITERATIVE REPAIR FOR ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy self-distillation (OPSD) supplies dense token-level feedback on trajectories sampled from the student's own policy, a richer training signal than the outcome-level rewards of reinforcement learning. This feedback comes from a teacher conditioned on a full reference solution unavailable to the student. The reference solution specifies the target but not how to move from the student's current error toward it, creating a *solution-conditioned shortcut risk*. We introduce AIR-OPD, an adaptive iterative repair framework for on-policy distillation that provides error-to-repair supervision. Given a failed response, a guidance generator synthesizes repair guidance for the current error. The student samples an on-policy retry with this guidance. If the retry remains incorrect, the generator produces new repair guidance for the newly observed error. At each round, a fixed teacher receives the guidance as privileged context and supervises the student on an error-aligned region of its latest failed response. Outcome-aware stage weighting favors early repair stages and credits stages whose immediate retry passes verification. We train AIR-OPD on the DAPO-Math-17K dataset and evaluate on AIME24, AIME25, and HMMT25, alongside out-of-distribution tests on MMLU-Pro and GPQA. We examine two guidance sources, self-guidance from the current student policy and external guidance from a larger model. For both Qwen3-4B and Qwen3-8B, AIR-OPD attains the best mathematical-reasoning averages, improving over the strongest baseline by up to 3.6 points, while preserving base-model performance on the out-of-distribution benchmarks.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) is now a standard approach to improving language-model reasoning in post-training (Shao et al., 2024; Yu et al., 2025). However, typical RLVR algorithms such as GRPO (Guo et al., 2025; Shao et al., 2024) rely on outcome rewards to compute response-level advantages for policy optimization. Such response-level supervision does not distinguish fine-grained token-level contributions within a trajectory. On-policy distillation (OPD) complements this outcome signal with dense token-level supervision on the student's own trajectories (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). On-policy self-distillation (OPSD) further removes the need for a separate teacher model (Zhao et al., 2026). Through in-context learning, the model serves as its own teacher, conditioned on a reference solution as privileged context that the student never observes.

Learning with privileged information uses training-only signals to guide the learner (Vapnik & Vashist, 2009; Lopez-Paz et al., 2016). In reference-conditioned OPSD, a successful attempt already satisfies the verifier, and further conditioning on a static reference may steer the student toward the reference's solution path. Privileged supervision is thus most informative on failed attempts, and recent work treats successful and failed trajectories asymmetrically (Hou et al., 2026; Zheng et al., 2026). For failed trajectories, a reference solution identifies the correct destination but does not specify the transition from the student's current error to a valid continuation. When the teacher remains conditioned on that same full solution, the student may learn to imitate privileged demonstrations rather than acquire generalizable error-correction mechanisms. We refer to this as the *solution-conditioned shortcut risk*.

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.

This leads to our central research question:

> *How can privileged supervision turn a failed on-policy trajectory into a sequence of learnable error-to-repair transitions?*

To answer this question, we introduce AIR-OPD. Inspired by instructional scaffolding (Wood et al., 1976), AIR-OPD performs multi-round adaptive repair that replaces a static reference-conditioned target with repair guidance synthesized for the student's latest error. Given a failed response, a privileged guidance generator uses the reference solution to synthesize repair guidance as scaffolding for the identified error, and the student samples an on-policy retry with this guidance. If the retry remains incorrect, the generator synthesizes fresh repair guidance targeting the newly observed error. At every round, the guidance serves as teacher-only privileged context for supervising the student on its latest failed response. AIR-OPD supervises only an error-aligned repair region of that response and excludes tokens that the repair guidance does not target, which would otherwise add noise. Each round thus contributes one stage loss. To combine these losses across rounds, we design outcome-aware stage weighting, which favors earlier repair stages and rewards stages whose retry passes verification.

We train on mathematical problems from DAPO-Math-17K (Yu et al., 2025), pairing each problem with a reference solution generated by GPT-5.5, and evaluate on AIME24, AIME25, and HMMT25. We additionally test transfer beyond this mathematical training distribution on MMLU-Pro and GPQA, and compare two guidance sources, self-guidance from the current student policy and external guidance from a larger model. Across these settings, AIR-OPD attains the best mathematical-reasoning averages on both Qwen3-4B and Qwen3-8B while preserving base-model performance on the out-of-distribution (OOD) benchmarks.

Our contributions are:

1. We introduce AIR-OPD, an adaptive iterative repair framework that aligns privileged on-policy supervision with the student's evolving errors. AIR-OPD synthesizes error-specific repair guidance and updates it in response to subsequent failed on-policy attempts.
2. We develop an error-aligned distillation objective that uses newly synthesized guidance as privileged teacher context when scoring the current failed response. Each supervision region covers the span where the teacher still expresses correction, and outcome-aware stage weighting allocates supervision across repair stages.
3. Across Qwen3-4B and Qwen3-8B with self- and external guidance, AIR-OPD improves average mathematical-reasoning accuracy by up to 3.6 percentage points over the strongest evaluated baseline, while keeping average OOD performance near that of the base model. Ablations support the roles of error-aligned supervision and stage weighting.

2 Related Work

Reinforcement learning with verifiable rewards. Building on reinforcement learning from human feedback (Schulman et al., 2017; Ouyang et al., 2022), reinforcement learning with verifiable rewards (RLVR) improves language-model reasoning using rewards from automatically checkable outcomes (Lambert et al., 2025; Guo et al., 2025; Yu et al., 2025). DeepSeekMath introduces Group Relative Policy Optimization (GRPO) for mathematical reasoning (Shao et al., 2024), while DAPO advances large-scale reasoning training through improved sampling and policy optimization (Yu et al., 2025). Process-level rewards densify this outcome signal by scoring intermediate steps (Lightman et al., 2024; Wang et al., 2024a), yet the supervision remains a scalar reward rather than a token-level target. For group-relative methods such as GRPO, difficult samples on which all rollouts are incorrect provide no relative advantage signal for updating the policy. To address this limitation, hint-based RL uses auxiliary guidance to make hard problems more tractable and recover useful training signals (Cao et al., 2026; Fan et al., 2026). StepHint guides exploration with partial solution steps (Zhang et al., 2026a). NuRL injects abstract hints when independent attempts fail (Chen et al., 2026), while Scaf-GRPO provides progressively more detailed hints from a pre-generated, fixed hint hierarchy (Zhang et al., 2026b). HiLL further learns hints conditioned on the reasoner's current failures, seeking to restore informative reward variation while encouraging transfer to no-hint reasoning (Xia et al., 2026). While these approaches use hints to reduce effective task difficulty and recover outcome-based

reward signals (from fixed pre-generated hierarchies or from a learned hinter), AIR-OPD synthesizes repair guidance for each failed response during multi-turn repair and uses it as privileged teacher context for token-level supervision of the repair.

On-policy (self-) distillation. Knowledge distillation trains a student to match a teacher's output distribution (Hinton et al., 2015; Kim & Rush, 2016). On-policy distillation (OPD) instead has the teacher score responses sampled from the student's own policy (Ross et al., 2011; Lin et al., 2020; Agarwal et al., 2024; Gu et al., 2024; Ko et al., 2024; Lu & Thinking Machines Lab, 2025), avoiding the mismatch between fixed demonstrations and the states the student reaches at deployment. OPSD extends OPD through in-context self-distillation, where the same model produces teacher and student distributions under asymmetric contexts (Zhao et al., 2026). In the reference-conditioned setting, the teacher receives a reference solution that is unavailable to the student, yielding dense token-level feedback on states that the student actually visits. Subsequent methods adjust the strength or routing of this signal using outcomes, reliability, or trajectory structure (Hou et al., 2026; Zheng et al., 2026; Zhang et al., 2026c). Closer to our setting, several distillation methods anchor supervision at failures (Xu et al., 2026; Wang et al., 2026). HINT-SD distills a feedback-conditioned teacher on failure-relevant spans selected in hindsight (Yeo et al., 2026), and TAPO inserts a diagnosis-and-correction transition after the learner's first critical error (Huang et al., 2026). Across these methods, the corrected trajectory or privileged context is constructed in a single pass and does not respond to how the learner fails again. We focus on how privileged context should be updated from one failed retry to the next.

Self-correction and iterative refinement. Inference-time methods treat an initial response as an intermediate state and use subsequent feedback to revise it. Self-Refine alternates self-generated feedback and refinement (Madaan et al., 2023), while Reflexion converts task feedback into verbal reflections that guide later trials (Shinn et al., 2023). Their effectiveness depends on the information in the feedback, as language models struggle to identify and repair complex reasoning mistakes without external reference signals (Huang et al., 2024).

Training-time methods instead seek to internalize improvement behavior. Self-training bootstraps from the model's own verified rationales (Zelikman et al., 2022; Singh et al., 2024; Hosseini et al., 2024), RISE fine-tunes on multi-turn revision trajectories (Qu et al., 2024), SCoRe learns multi-turn self-correction under the model's own distribution via reinforcement learning (Kumar et al., 2025), and SuperCorrect distills teacher-generated thought templates and correction traces (Yang et al., 2025). AIR-OPD uses iterative repair to construct error-aligned distillation targets on the student's own failed responses: at each stage, a teacher conditioned on newly synthesized repair guidance supervises the current response.

3 PRELIMINARIES: ON-POLICY SELF-DISTILLATION

Let q denote a problem and p_θ a trainable student policy that defines next-token distributions given a conditioning context. On-policy distillation (OPD) trains the student on states that it actually visits (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Given a response $y \sim p_\theta(\cdot \mid q)$ sampled from the current policy, a teacher p_{T} scores every token y_j given its prefix $y_{<j}$ (tokens indexed from 0), and OPD minimizes the token-level divergence

$$\mathcal{L}_{\mathrm{OPD}}(q, y) = \frac{1}{|y|} \sum_{j=0}^{|y|-1} D\big(p_{\mathrm{T}}(\cdot \mid q, y_{<j}) \,\big\|\, p_\theta(\cdot \mid q, y_{<j})\big)\,. \tag{1}$$

Because the response is sampled from p_θ, the teacher provides dense feedback on prefixes generated by the evolving student. In OPD, the teacher is typically a separate, more capable model.

On-policy self-distillation (OPSD) further initializes the student and the teacher from the same model, removing the need for a separately trained teacher (Zhao et al., 2026). Providing the reference solution z as additional input to the teacher gives it information that the student cannot access. Here the teacher p_{T} is instantiated as the frozen initial backbone $\bar{p}$. For an on-policy response $y \sim p_\theta(\cdot \mid q)$, distillation compares the two branches at the same student-generated prefixes $y_{<j}$. The teacher

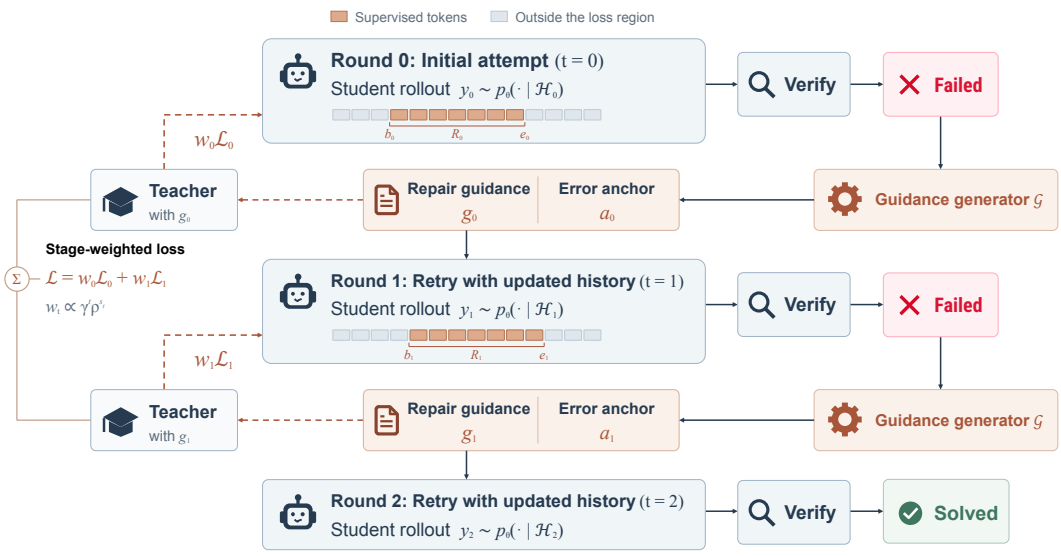


Figure 1: **Overview of AIR-OPD.** When an on-policy rollout fails verification, the guidance generator $\mathcal{G}$ quotes the first flawed step as an error anchor a_t and synthesizes repair guidance g_t. A frozen teacher conditioned on g_t then supervises the student over the error-aligned repair region R_t, and the student retries with the guidance in context. A verified retry ends the trajectory, and outcome-aware stage weighting combines the stage losses with weights $w_t \propto \gamma^t \rho^{s_t}$.

conditions on $(q, z, y_{<j})$, the student conditions on $(q, y_{<j})$, and the OPSD loss is

$$\mathcal{L}_{\mathrm{OPSD}}(q, z, y) = \frac{1}{|y|} \sum_{j=0}^{|y|-1} D\big(p_{\mathrm{T}}(\cdot \mid q, z, y_{<j}) \,\big\|\, p_\theta(\cdot \mid q, y_{<j})\big)\,, \tag{2}$$

where only the teacher branch observes z. Following OPSD, we instantiate D as the forward KL divergence $D_{\mathrm{KL}}(p_{\mathrm{T}} \,\|\, p_\theta)$ and apply pointwise clipping, which caps each vocabulary entry's divergence contribution so that a few high-divergence tokens cannot dominate the update.

In OPSD, in-context learning thus converts the reference solution into fine-grained supervision over the probability of each token in the student's on-policy response. However, under this construction, the teacher must extract relevant information from the complete reference solution and relate it to the student's current reasoning response. This process is performed implicitly and does not explicitly teach the student how to transition from its current error state to a valid solution state.

4 AIR-OPD: ADAPTIVE ITERATIVE REPAIR FOR ON-POLICY DISTILLATION

AIR-OPD makes explicit the extraction step that OPSD leaves to the teacher (Figure 1). A guidance generator reads the reference solution and writes repair guidance for the student's current error, and the teacher receives this guidance in place of the full reference solution. The rest of the method answers three questions: how the guidance is updated across repair rounds (Section 4.1), which tokens of each failed response the teacher supervises (Section 4.2), and how the resulting stage losses are combined (Section 4.3).

4.1 MULTI-ROUND ADAPTIVE REPAIR

AIR-OPD performs up to K repair rounds, adapting the guidance to the student's latest failed response. Let $\mathcal{H}_t$ denote the student-visible dialogue history at stage t, with $\mathcal{H}_0$ containing only the problem q. The student generates an on-policy response

$$y_t \sim p_\theta(\cdot \mid \mathcal{H}_t). \tag{3}$$

If y_t fails verification by an outcome verifier V, the guidance generator $\mathcal{G}$ uses the reference solution z to diagnose the current failure and produce

$$(a_t, g_t) = \mathcal{G}(q, z, \mathcal{H}_t, y_t). \tag{4}$$

Here, a_t quotes the identified first flawed step verbatim, and g_t provides concise guidance for correcting it. Rule-based checks filter out guidance that reveals the final answer (Appendix B details the rules).

To advance the repair process, we prompt the student to repair the previous erroneous response y_t according to g_t:

$$\mathcal{H}_{t+1} = \mathrm{Update}(\mathcal{H}_t, y_t, g_t). \tag{5}$$

Update appends y_t to the history as the model's own output turn, and appends g_t, rendered with a repair instruction, as the next input turn. If budget remains, the student then generates the retry $y_{t+1} \sim p_\theta(\cdot \mid \mathcal{H}_{t+1})$; if the retry remains incorrect, $\mathcal{G}$ re-localizes the error in y_{t+1} and generates updated guidance. The process ends when a response passes verification or the budget is exhausted. Throughout this process, the privileged context keeps adapting to the student's current error state. Appendix D.1 presents worked repair examples from training.

4.2 Error-aligned privileged distillation

At stage t, the teacher and student score the same failed response y_t under the shared dialogue history $\mathcal{H}_t$. The frozen teacher p_{T} additionally receives the newly generated guidance g_t, which is withheld from the student when scoring y_t.

To cover both the current error and the potentially recoverable region that follows, the supervision region starts at the beginning of the error region. We exactly match a_t against y_t to obtain the start position b_t. Supervising the error region alone would lose this recoverable region, so we rely on the corrective signal from the teacher to determine the endpoint of the supervision region. At position j, we measure this signal as the total positive teacher–student probability difference over alternatives to the observed token:

$$C_{t,j} = \sum_{v \neq y_{t,j}} \left[p_{\mathrm{T}}(v \mid \mathcal{H}_t, g_t, y_{t,<j}) - p_\theta(v \mid \mathcal{H}_t, y_{t,<j}) \right]_+ . \tag{6}$$

Both distributions are already computed for the stage loss, so measuring this signal introduces no additional computational overhead.

To avoid ending the region within a reasoning block, we partition the suffix of y_t starting at b_t at blank-line boundaries. Let $\mathcal{E}_t$ denote the resulting block endpoints, including the response end $|y_t|$. For a tolerance $\epsilon \in (0, 1)$, we select the earliest endpoint retaining at least a $1 - \epsilon$ fraction of the suffix's total corrective signal:

$$e_t = \min\Big\{ r \in \mathcal{E}_t : \textstyle\sum_{j=b_t}^{r-1} C_{t,j} \geq (1-\epsilon) \sum_{j=b_t}^{|y_t|-1} C_{t,j} \Big\}. \tag{7}$$

By construction, the discarded tail carries at most an ϵ fraction of the suffix's total corrective signal. If the total signal is zero, we retain the full suffix. Following Eq. 2, the stage loss averages the divergence over the selected repair region $R_t = y_t[b_t : e_t]$, i.e., tokens b_t to $e_t - 1$:

$$\mathcal{L}_t = \frac{1}{e_t - b_t} \sum_{j=b_t}^{e_t-1} D\big(p_{\mathrm{T}}(\cdot \mid \mathcal{H}_t, g_t, y_{t,<j}) \,\big\|\, p_\theta(\cdot \mid \mathcal{H}_t, y_{t,<j})\big) . \tag{8}$$

Gradients flow only through the student branch.

4.3 Outcome-aware stage weighting

Training samples contain different numbers of repair stages, and directly summing stage losses would grant samples with more repair rounds a larger total weight. To avoid this, we introduce *outcome-aware stage weighting*. The scheme allocates the stage weights within each supervised trajectory by stage depth and the outcome of the retry immediately following each stage.

This allocation rests on two assumptions about the repair process. First, imperfect retries accumulate noise across stages. The later a stage is, the more failed responses and repair guidance accumulate in

its context and the less reliable its supervision signal becomes, so the weights should lean toward earlier stages. Second, whether the immediate retry passes verification reflects the stage's supervision quality. A successful retry indicates that the stage's repair guidance pointed to a valid repair, and such a stage should receive an additional weight bonus.

Concretely, for trajectory n with final supervised stage T_n and stage losses $\mathcal{L}_{n,t}$ (Eq. 8), let $s_{n,t} = \mathbb{1}[V(y_{n,t+1}) = 1]$ indicate whether the on-policy retry immediately following stage t passes the outcome verifier V, with $s_{n,t} = 0$ when no retry is available (after the final round). The stage weights are

$$w_{n,t} = \frac{\widetilde{w}_{n,t}}{\sum_{k=0}^{T_n} \widetilde{w}_{n,k}}, \qquad \widetilde{w}_{n,t} = \gamma^t \rho^{s_{n,t}}, \\ \gamma \in (0, 1], \qquad \rho \geq 1. \tag{9}$$

The temporal factor γ^t (with $\gamma < 1$) decays the weights with stage depth, the outcome bonus $\rho^{s_{n,t}}$ rewards a verified immediate retry, and the normalization gives each supervised trajectory unit total weight.

For a minibatch of repair trajectories $\mathcal{B}$, the final objective is

$$\mathcal{L}_{\text{AIR-OPD}} = \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \sum_{t=0}^{T_n} w_{n,t} \mathcal{L}_{n,t}. \tag{10}$$

The complete algorithm is summarized in Appendix A.

5 Experimental Setup

5.1 Models, training data, and baselines

We use two Qwen variants of different scales, Qwen3-4B and Qwen3-8B (Qwen Team, 2025). We train on the DAPO-Math-17K dataset (Yu et al., 2025), generating the reference solution for each problem with GPT-5.5 and retaining the 15,650 problems whose generated solution verifies against the dataset answer. Outcome verification at every repair round uses rule-based answer matching. We compare *self-guidance* generated by the current student policy with *external guidance* generated by a larger model (Qwen3.5-27B). The main baselines are the corresponding base model and three trained methods. Supervised fine-tuning (SFT) trains on the reference solutions with a cross-entropy loss. GRPO optimizes rule-based outcome rewards over groups of on-policy rollouts. OPSD distills a reference-conditioned teacher on the student's own rollouts. All trained baselines are our reproductions on the same training corpus and backbones (Appendix C).

5.2 Evaluation benchmarks and metrics

We evaluate models along two dimensions, mathematical reasoning and out-of-distribution generalization. Mathematical reasoning is evaluated on AIME24, AIME25, and HMMT25 (HMMT, 2025). Each mathematical benchmark contains 30 problems. We report per-benchmark accuracy as Avg@12, with *Math Avg.* the unweighted mean of the three benchmark scores. Out-of-distribution evaluation uses MMLU-Pro (Wang et al., 2024b) and GPQA (Rein et al., 2024) to assess transfer beyond the mathematical training distribution, reporting accuracy as Avg@4, with *OOD Avg.* the mean of the two benchmarks. During evaluation, no repair guidance is added to the model input.

5.3 Implementation details

We adopt the OPSD training recipe (Zhao et al., 2026), using the frozen initial backbone with its adapters disabled as the distillation teacher. During training, the student operates in non-thinking mode and the teacher in thinking mode. Optimization uses rank-64 LoRA adapters (Hu et al., 2022) and AdamW (Loshchilov & Hutter, 2019) with a learning rate of 5×10^{-6} and an effective batch size of 32.

AIR-OPD is built on TRL (von Werra et al., 2020) and trains for 100 optimizer steps. Each problem allows at most $K = 3$ adaptive repair rounds, and each round samples a single on-policy rollout

Table 1: **Main results on mathematical-reasoning (Avg@12) and out-of-distribution (Avg@4) benchmarks.** Both AIR-OPD variants surpass all baselines on Math Avg. at each scale while preserving base-model performance on the OOD group. SELF-G/EXTERNAL-G generate repair guidance with the current student policy and Qwen3.5-27B, respectively. **Bold**/underlined: best/second-best within each block.

	Mathematical Reasoning				Out-of-Distribution		
Method	AIME24	AIME25	HMMT25	Math Avg.	MMLU-Pro	GPQA	OOD Avg.
Qwen3-4B							
Base model	75.0	66.4	42.2	61.2	51.5	**62.9**	**57.2**
SFT	70.6	61.4	42.8	58.2	43.1	52.0	47.6
GRPO	75.6	67.5	43.9	62.3	51.3	62.4	56.8
OPSD	76.4	69.4	43.3	63.1	50.4	60.2	55.3
AIR-OPD (SELF-G)	76.7	73.1	**47.8**	65.8	**52.5**	61.4	56.9
AIR-OPD (EXTERNAL-G)	**80.0**	**73.3**	46.7	**66.7**	51.7	61.9	56.8
Qwen3-8B							
Base model	75.8	65.6	43.9	61.8	57.9	64.1	61.0
SFT	75.3	63.6	43.6	60.8	46.5	55.8	51.2
GRPO	76.9	69.2	45.8	64.0	56.6	**64.5**	60.6
OPSD	77.8	69.7	46.7	64.7	57.4	61.7	59.6
AIR-OPD (SELF-G)	78.3	72.5	**49.7**	66.9	**58.3**	63.5	60.9
AIR-OPD (EXTERNAL-G)	**79.7**	**73.1**	49.4	**67.4**	58.0	64.1	**61.1**

from the current student (temperature 1.1, top-p 0.95), capped at 8,192 tokens. Sampler weights are synchronized after every optimizer step. The default stage weighting is $(\gamma, \rho) = (0.8, 1.5)$, and the endpoint budget is $\epsilon = 0.1$. Both guidance sources are generated in thinking mode (temperature 0.7, top-p 0.9) and filtered by the same validation rules. All experiments are conducted on a single node with eight H100 GPUs. Appendix C documents the complete training and evaluation configuration.

6 MAIN RESULTS

AIR-OPD achieves the best mathematical reasoning at both scales. Table 1 compares AIR-OPD against the base model, SFT, GRPO, and OPSD. Both guidance settings surpass every baseline on Math Avg., and external guidance, where a stronger generator supplies higher-quality repair guidance, exceeds the strongest baseline, OPSD, by 3.6 and 2.7 points at 4B and 8B, respectively. The improvement is consistent across benchmarks. AIR-OPD variants attain the best score on every mathematical benchmark at both scales, with pronounced gains on HMMT25.

Self-guidance alone accounts for most of the improvement. Without introducing any external model, the self-guided variant already improves over OPSD by 2.8 and 2.1 Math Avg. points. External guidance further improves the Math Avg. The additional gain from external guidance is smaller at 8B than at 4B, possibly because the larger student generates better repair guidance itself.

AIR-OPD attains stronger mathematical reasoning while preserving near-base OOD performance. SFT lowers the OOD average by 9.6 and 9.8 points at 4B and 8B, consistent with forgetting under task-specific fine-tuning (Kirkpatrick et al., 2017; Kotha et al., 2024), and OPSD also regresses at both scales. In contrast, AIR-OPD stays within 0.4 points of the base model at 4B and within 0.1 points at 8B. It is the only evaluated reference-based training method that maintains a near-base OOD average at both scales. GRPO preserves OOD performance comparably, but its Math Avg. trails AIR-OPD by 2.9–4.4 points. One possible explanation lies in the granularity of supervision. RLVR provides only scalar feedback, and full-response distillation supervises every token, whereas AIR-OPD provides corrective signals only on regions aligned with the student's errors. This selective supervision may limit over-specialization to the mathematical training distribution while preserving the gains of distillation.

7 Ablations and Analysis

All ablations start from the default Qwen3-4B configuration of Section 5.3 and report the in-domain Math Avg. We organize the analysis into three parts. Section 7.1 controls for failure-only sample selection and ablates the supervision region and the stage weighting of the objective, Section 7.2 measures how large a repair budget is needed, and Section 7.3 traces how capability evolves over the course of training relative to GRPO and OPSD.

7.1 Effect of sample selection, supervision region, and stage weighting

In Table 2, the two OPSD rows vary which rollouts are supervised, and each remaining row changes one component of AIR-OPD, the supervision region (Section 4.2) or the stage weighting (Section 4.3), while keeping the rest of the default configuration.

Sample selection. AIR-OPD supervises only failed rollouts, so its gains could stem from this selection alone. Applying the OPSD loss only to failed rollouts yields 62.5, 0.6 points below standard OPSD and 3.3 below AIR-OPD. The gains thus depend on how AIR-OPD supervises each failure, not on which rollouts it selects.

Table 2: **Ablation results (Qwen3-4B, SELF-G).** We test failure-only sample selection for OPSD and ablate the supervision region and the stage-weighting components of AIR-OPD.

Setting	Math Avg.
AIR-OPD	65.8
Sample selection (OPSD)	
All rollouts	63.1
Failed rollouts only	62.5
Supervision region	
w/o error alignment	64.7
Error block only	64.2
Stage weighting (γ, ρ)	
Uniform $(1,\ 1)$	65.0
Temporal factor only $(0.8,\ 1)$	65.4
Outcome bonus only $(1,\ 1.5)$	65.4

Supervision region. We compare the default repair region R_t with two ablations. *W/o error alignment* supervises every token of the failed response in place of this region, and *error block only* ends the region at the boundary of the first reasoning block. Both ablations lower the Math Avg. Widening supervision to the full response costs 1.1 points, and narrowing it to the error block alone costs 1.6. We suspect that full-response supervision dilutes the signal with tokens the repair guidance does not target, while the error block alone omits downstream reasoning that may also require correction. These results support supervising both the identified error region and the selected downstream reasoning.

Stage weighting. We ablate the temporal factor γ^t and the outcome bonus $\rho^{s_{n,t}}$ of Eq. 9. *Uniform* $(1, 1)$ removes both factors and weights all stages equally, while *temporal factor only* $(0.8, 1)$ and *outcome bonus only* $(1, 1.5)$ each restore a single factor. Either factor alone already improves over uniform weighting, and combining the two performs best. This indicates that the weights should combine a preference for earlier stages with feedback from the immediate retry outcome rather than allocate supervision by a single criterion.

7.2 Effect of the repair budget

We vary the maximum number of repair rounds $K \in \{1, \ldots, 5\}$ for both guidance settings. Figure 2 traces Math Avg. as the budget grows alongside the number of supervised stages. Both variants already surpass OPSD with a single repair round, and the overall trend rises and then flattens. At $K = 1$ every supervised trajectory contains exactly one stage, so this gain does not depend on additional repair rounds. $K = 1$ also exceeds failure-only OPSD by

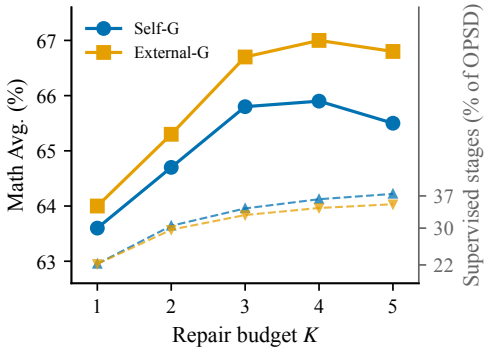


Figure 2: **Effect of the repair budget K (Qwen3-4B).** Math Avg. (left axis) rises through the default $K = 3$ and then flattens, while each variant's number of supervised stages (dashed, right axis; % of the samples OPSD supervises) grows with diminishing increments.

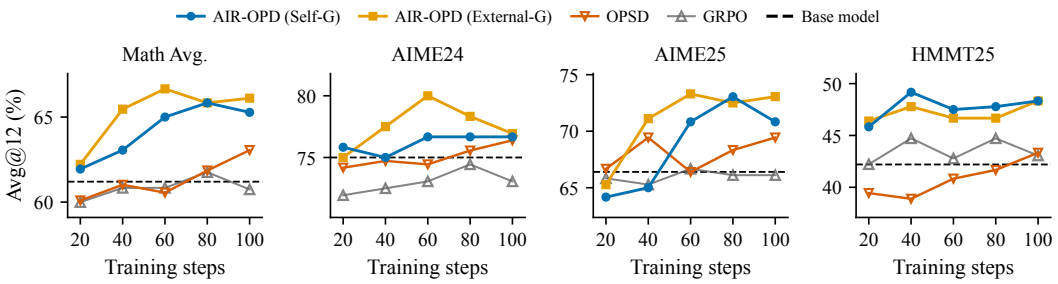


Figure 3: **Training dynamics on the mathematical benchmarks (Qwen3-4B).** Both AIR-OPD variants lead OPSD and GRPO on the three-benchmark average from the first checkpoint onward. Curves show Avg@12 at checkpoints saved every 20 optimizer steps, and dashed lines mark the base model.

1.1–1.5 points (Table 2), suggesting that error-aligned repair supervision provides a stronger learning signal than reference-conditioned full-response distillation. Raising the budget to $K = 3$ brings further gains, which confirms the value of continuing to supply fresh repair signals for unresolved failures during training. Beyond this point both curves show diminishing returns. $K = 4$ adds only a small improvement, and $K = 5$ falls slightly below $K = 4$. We suspect that stages added at deep rounds carry accumulated noise, as their contexts pile up failed attempts and guidance, and their less reliable supervision begins to offset the gains.

The dashed curves show that the number of supervised stages keeps increasing with K but with shrinking increments, as later rounds activate only for trajectories that remain incorrect and yield validated repair guidance rather than for all trajectories. The count rises from 22.2% of the samples OPSD supervises at $K = 1$ to 35.2–37.4% at $K = 5$. We keep $K = 3$ as the practical default. It captures most of the observed gains while requiring fewer supervised stages than larger budgets. Comparing the two dashed curves further shows that external guidance adds fewer supervised stages per extra round than self-guidance, possibly because higher-quality external guidance resolves more failures in earlier rounds.

7.3 Training dynamics

Figure 3 plots the three-benchmark average and per-benchmark accuracy over the checkpoints of both AIR-OPD variants, OPSD, and GRPO, saved every 20 optimizer steps, with the base model as a fixed reference. On the three-benchmark average, both AIR-OPD variants lead both baselines at every checkpoint, and external guidance matches or exceeds the self-guided average throughout. AIR-OPD also converges substantially faster. By step 40, both variants already reach the score OPSD attains at step 100. This faster convergence also uses much less supervision. At the default $K = 3$, the number of stages supervised by AIR-OPD is approximately one third of the number of samples supervised by OPSD (Section 7.2), with each stage covering an error-aligned repair region rather than a full response. This suggests that supervision concentrated on erroneous regions yields more learning signal per supervised token.

8 Conclusion

We introduced AIR-OPD, an adaptive iterative repair framework for on-policy distillation that addresses the solution-conditioned shortcut risk of reference-privileged supervision. Instead of conditioning the teacher on a static reference solution, a guidance generator synthesizes answer-free repair guidance that evolves with the student's errors, the teacher distills each repair over an error-aligned region of the latest failed response, and outcome-aware stage weighting allocates supervision across rounds. Across two model scales, both guidance sources surpass strong RL and distillation baselines on mathematical reasoning while preserving base-model performance out of distribution, and self-guidance alone accounts for most of the improvement. Our analysis further shows that the gains emerge from the first checkpoints of training, and that a small repair budget captures most of the gains observed in our sweep while supervising roughly a third as many samples as OPSD.

AI USE STATEMENT

Among the tasks that require disclosure, we used generative AI tools to assist with translation and with implementing the method. The authors reviewed and tested AI-generated code before use. Generative models are also a component of the method itself, and the reference solutions for training were generated by GPT-5.5. We did not use generative AI tools to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or give feedback on the research methodology or the experiments, or to clean or reformat the dataset. Tasks concerning mathematical claims, proofs, and qualitative or thematic analysis do not apply to this work. Among the tasks with recommended disclosure, we used generative AI tools to search for and summarize related literature and to draft and edit parts of the manuscript for readability. We have reviewed all AI-assisted work and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

To support reproducibility, we specify the method and the experimental protocol in full. Section 4 defines the multi-round repair loop, the error-aligned stage loss, and the outcome-aware stage weighting, and Section 5 specifies the training corpus, the baselines, the benchmarks, and the evaluation protocol. Appendix A gives the complete training algorithm, Appendix B reproduces the guidance prompt templates and the validation rules, and Appendix C documents the training and evaluation configurations for AIR-OPD and every baseline. Appendix D provides worked repair examples drawn from the training logs. Our implementation, together with the training and evaluation configurations and the repair-guidance prompt bundle, is available at `https://anonymous.4open.science/r/air-opd-1D09`.

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. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=3zKtaqxLhW`.

Zhiyu Cao, Kaixin Wu, Mingjie Zhong, Peifeng Li, Xiaobo Li, Can Ye, and Qiaoming Zhu. Hint-guided diversified policy optimization for LLM reasoning. *arXiv preprint arXiv:2606.03021*, 2026. URL `https://arxiv.org/abs/2606.03021`.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=hfNnQHkTtv`.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=mZn2Xyh9Ec`.

Yuchun Fan, Bei Li, Peiguang Li, Yilin Wang, Yongyu Mu, Jian Yang, Xin Chen, Rongxiang Weng, Jingang Wang, Xunliang Cai, Jingbo Zhu, and Tong Xiao. LANG: Reinforcement learning for multilingual reasoning with language-adaptive hint guidance. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43838–43866, 2026. URL `https://aclanthology.org/2026.acl-long.2029/`.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=5h0qf7IBZZ`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL `https://doi.org/10.1038/s41586-025-09422-z`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL `https://arxiv.org/abs/1503.02531`.

HMMT. Harvard–MIT mathematics tournament (HMMT) february 2025. https://www.hmmt.org/, 2025. Accessed 2026-09.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, Chengquan Zhang, Zhuotao Tian, Han Hu, Yi Yang, Fei Wu, and Hehe Fan. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026. URL https://arxiv.org/abs/2605.03677.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Zhilin Huang, Hang Gao, Ziqiang Dong, Yuan Chen, Yifeng Luo, Chujun Qin, Jingyi Wang, Yang Yang, and Guanjun Jiang. Learning from your own mistakes: Constructing learnable micro-reflective trajectories for self-distillation. *arXiv preprint arXiv:2606.18844*, 2026. URL https://arxiv.org/abs/2606.18844.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1317–1327, 2016. URL https://aclanthology.org/D16-1139/.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/ko24c.html.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VrHiF2hsrm.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w. Oral.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6121–6133, 2020. URL https://aclanthology.org/2020.emnlp-main.494/.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1511.03643.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Lu and Thinking Machines Lab. On-policy distillation. Thinking Machines Lab: Connectionism, 2025. URL https://thinkingmachines.ai/blog/on-policy-distillation/. Published Oct 27, 2025; doi:10.64434/tml.20251026.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.17651.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://openreview.net/forum?id=DRC9pZwBwR.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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 *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/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 (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.11366.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Bing Wang, Shaotian Yan, Chen Shen, Kaiyuan Liu, Sinan Fan, Ximing Li, Rui Miao, Xiaosong Yuan, Zhanming Shen, and Jieping Ye. Backtracking when it strays: Mitigating dual exposure biases in LLM reasoning distillation. *arXiv preprint arXiv:2605.19433*, 2026. URL https://arxiv.org/abs/2605.19433.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a. URL https://aclanthology.org/2024.acl-long.510/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

David Wood, Jerome S. Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976. doi: 10.1111/j.1469-7610.1976.tb00381.x.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026. URL https://arxiv.org/abs/2604.00698.

Haolei Xu, Xiaowen Xu, Haiwen Hong, Zixuan Ni, Hongxing Li, Yiwen Qiu, Weiming Lu, and Yongliang Shen. Pass the baton: Trajectory-relayed on-policy distillation. *arXiv preprint arXiv:2607.26057*, 2026. URL https://arxiv.org/abs/2607.26057.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=PyjZO7oSw2.

Woongyeong Yeo, Yumin Choi, Taekyung Ki, and Sung Ju Hwang. HINT-SD: Targeted hindsight self-distillation for long-horizon agents. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026. URL https://arxiv.org/abs/2605.17873. arXiv:2605.17873.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html.

Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. StepHint: Multi-level stepwise hints enhance reinforcement learning to reason. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37846–37864, 2026a. doi: 10.18653/v1/2026.acl-long.1755. URL https://aclanthology.org/2026.acl-long.1755/.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-GRPO: Scaffolded group relative policy optimization for enhancing LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=bOwVr0yr7r.

Yanfei Zhang, Xu Lin, and Chenglin Wu. StepOPSD: Step-aware online preference self-distillation for agent reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026c. URL https://arxiv.org/abs/2605.27140. arXiv:2605.27140.

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. *arXiv preprint arXiv:2601.18734*, 2026. URL https://arxiv.org/abs/2601.18734.

Binbin Zheng, Xing Ma, Yiheng Liang, Jingqing Ruan, Xiaoliang Fu, Kepeng Lin, Benchang Zhu, Ke Zeng, and Xunliang Cai. Scope: Signal-calibrated on-policy distillation enhancement with dual-path adaptive weighting. *arXiv preprint arXiv:2604.10688*, 2026. URL https://arxiv.org/abs/2604.10688.

A Complete Training Algorithm

Algorithm 1 gives the complete procedure that AIR-OPD runs on one training minibatch, from the initial on-policy rollouts to the returned training loss. The method definitions and design rationale remain in Section 4.

Algorithm 1 AIR-OPD: multi-round repair and stage-weighted distillation

Input: minibatch $\mathcal{B}$ of problems q with reference solutions z, student p_θ, frozen teacher p_{T}, outcome verifier V, and guidance generator $\mathcal{G}$
Hyperparameters: repair budget K, endpoint tolerance $\epsilon \in (0,1)$, temporal factor $\gamma \in (0,1]$, and outcome bonus $\rho \geq 1$
Output: minibatch training loss $\mathcal{L}_{\text{AIR-OPD}}$
for all $(q,z) \in \mathcal{B}$, indexed by n **do**
  $\mathcal{S}_n \leftarrow \varnothing$, $\mathcal{L}_n \leftarrow 0$, $t \leftarrow 0$, $\mathcal{H}_0 \leftarrow \mathrm{Render}(q)$, and sample $y_0 \sim p_\theta(\cdot \mid \mathcal{H}_0)$
  while $t < K$ and y_t is a well-formed response with $V(y_t) = 0$ **do**
    $(a_t, g_t) \leftarrow \mathcal{G}(q, z, \mathcal{H}_t, y_t)$
    if (a_t, g_t) fails validation **then**
      break
    end if
    $b_t \leftarrow$ the exact match position of a_t in y_t
    $e_t \leftarrow$ the earliest block endpoint with corrective coverage $1-\epsilon$
    Compute $\mathcal{L}_{n,t}$ on the repair region $R_t = y_t[b_t : e_t]$ and add $(t, \mathcal{L}_{n,t})$ to $\mathcal{S}_n$
    $\mathcal{H}_{t+1} \leftarrow \mathrm{Update}(\mathcal{H}_t, y_t, g_t)$ and, if $t < K-1$, sample $y_{t+1} \sim p_\theta(\cdot \mid \mathcal{H}_{t+1})$
    $s_{n,t} \leftarrow \mathbb{1}[V(y_{t+1}) = 1]$, or 0 if no retry is sampled, and $t \leftarrow t+1$
  end while
  if $\mathcal{S}_n \neq \varnothing$ **then**
    $T_n \leftarrow t-1$
    $w_{n,t} \leftarrow \gamma^t \rho^{s_{n,t}} / \sum_{k=0}^{T_n} \gamma^k \rho^{s_{n,k}}$ for $t = 0, \ldots, T_n$
    $\mathcal{L}_n \leftarrow \sum_{t=0}^{T_n} w_{n,t} \mathcal{L}_{n,t}$
  end if
end for
return $\mathcal{L}_{\text{AIR-OPD}} = \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \mathcal{L}_n$

Guidance generator — first-round user template

[Problem]
{problem}

[Private Correct Final Answer]
{private_answer}

[Reference Solution]
{private_reference_solution}

[QUOTE SOURCE - Incorrect Solution]
{incorrect_response}

Diagnose [QUOTE SOURCE - Incorrect Solution] using the Reference Solution and the Private Correct Final Answer as privileged checking evidence. First decide whether the quoted response is mathematically correct.

If the response is not mathematically correct, identify the earliest causal mathematical error that still affects the conclusion. The <error_quote> MUST contain only a short, verbatim, contiguous substring copied from [QUOTE SOURCE - Incorrect Solution]. Copy only the exact mathematical expression without conversational words or quotation marks. Do not quote the problem, reference solution, private correct final answer, or these instructions. If the first flawed step is unclear, quote the earliest local claim in the source whose logic should be checked.

The <repair_guidance> must identify the exact mistake, explain how to fix it, and state the correct governing logic. It must be problem-specific, actionable, answer-free, and must not reproduce a complete corrected solution or reveal the final answer.

When the reasoning is mathematically correct, please output no_repair.

Respond strictly in exactly one of these forms:

<decision>no_repair</decision>

or:

<decision>repair</decision>
<error_quote>A short exact substring copied only from the QUOTE SOURCE block</error_quote>
<repair_guidance>Identify the exact mistake in the quote, explain how to fix it, and state the correct logic without revealing the final answer</repair_guidance>

Figure 4: **Guidance generator, first repair round.** The template diagnoses the initial failed response y_0 under the privileged reference solution and final answer.

B Prompts and Validation Details

B.1 Guidance generator instruction

Figures 4 and 5 show the full user templates of the guidance generator for the first repair round and for the later rounds. At every round the generator receives the problem, the reference solution and its final answer as privileged checking evidence, and the latest failed response. From the second round it also receives the guidance the student has already seen. Braced fields are filled at runtime.

B.2 Repair dialogue

The retry is sampled from a multi-turn dialogue whose turns are shown below. The initial user turn poses the problem, and each unsuccessful response is followed by a feedback turn that delivers the current repair guidance. From the second feedback turn onward the opening line instead reads `The previous answer is still incorrect.`

Guidance generator — later-round user template (rounds 2–3)

```
[Problem]
{problem}
[Private Correct Final Answer]
{private_answer}
[Reference Solution]
{private_reference_solution}
[QUOTE SOURCE - Latest Response]
{latest_incorrect_response}
[Current Repair Guidance]
{current_repair_guidance}
```

Diagnose [QUOTE SOURCE - Latest Response] using the Reference Solution and the Private Correct Final Answer as privileged checking evidence. First decide whether the latest response is mathematically correct after the Current Repair Guidance has been applied. If the latest response is mathematically correct, please output `no_repair`.

If it is not mathematically correct, write a replacement repair guidance: identify the earliest remaining causal mathematical error that still affects the conclusion, correct only that bottleneck, and give exactly one next operation the solver can perform. Do not merely restate a correction that has already been executed. Preserve still-valid prior guidance and replace guidance that is now misleading.

The `<error_quote>` MUST contain only a short, verbatim, contiguous substring copied from [QUOTE SOURCE - Latest Response]. Copy only the exact mathematical expression without conversational words or quotation marks. Do not quote the problem, reference solution, private correct final answer, Current Repair Guidance, or these instructions. If the first remaining flaw is unclear, quote the earliest local claim in the latest source whose logic should be checked.

The `<repair_guidance>` must be problem-specific, actionable, answer-free, and must not reproduce a complete corrected solution or reveal the final answer. It must state the replacement correction needed at this round.

Respond strictly in exactly one of these forms:

```
<decision>no_repair</decision>
```

or:

```
<decision>repair</decision>
<error_quote>A short exact substring copied only from the latest QUOTE SOURCE
block</error_quote>
<repair_guidance>One concise replacement repair guidance for the remaining bottleneck, without
revealing the final answer</repair_guidance>
```

Figure 5: **Guidance generator, later repair rounds.** The template replaces the current guidance after a retry that is still incorrect, and additionally receives the guidance the student has already seen.

Repair dialogue — initial problem turn

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

Repair dialogue — feedback turn after an unsuccessful response

The previous answer is incorrect.

Use the following repair guidance, which explains how to correct the reasoning:
`=== Repair Guidance Begin ===`
{`current_repair_guidance`}
`=== Repair Guidance End ===`

Revise the reasoning and answer again. Do not simply repeat the previous attempt.

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

B.3 Validation rules

The generator output is accepted only when it matches one of the two declared forms in full, with a nonempty error quote and a nonempty repair-guidance field. The error quote must be located in the latest response, and guidance that states the final answer is rejected. The leak check excludes the error quote and rejects guidance containing boxed answers or rule-based matches to the normalized reference answer, such as explicit answer statements or variable assignments. These surface-form checks do not guarantee detection of all semantic answer leakage. When any check fails, or when the generator returns `no_repair`, the round yields no supervised stage and stops further retries.

C Training and Evaluation Details

C.1 Training settings

Following the comparison setup of OPSD (Zhao et al., 2026), Table 3 places SFT, GRPO, OPSD, and AIR-OPD side by side, grouping their optimization and on-policy sampling settings. All trained baselines in Table 1 are our own reproductions on the same training corpus and backbones as AIR-OPD; the GRPO reproduction applies no KL penalty, and the untrained base-model row has no optimization settings. Training problems come from DAPO-Math-17K (Yu et al., 2025), and outcome verification at every repair round uses rule-based answer matching. Reference solutions are generated by GPT-5.5 and retained only when their final answers verify against the dataset answers, leaving 15,650 of the 17,398 problems as training rows. SFT minimizes cross-entropy on these reference solutions without on-policy sampling and uses a 16,000-token maximum sequence length measured over the full supervised sequence.

All trained methods are implemented on top of TRL (von Werra et al., 2020) and use AdamW, BF16, and FlashAttention-2 (Dao, 2024), with a colocated vLLM sampler (Kwon et al., 2023) for on-policy rollouts. For OPSD and AIR-OPD, the pointwise divergence-clipping threshold is $\tau = 0.05$.

AIR-OPD-specific settings. AIR-OPD adds a repair loop and a guidance generator on top of the shared configuration in Table 3. Each problem allows at most $K = 3$ repair rounds, with stage weighting $(\gamma, \rho) = (0.8, 1.5)$ and endpoint tolerance $\epsilon = 0.1$. The student decodes with top-p 0.95 and top-k 20, generating at most 8,192 completion tokens under a total sequence-length limit of 19,168 tokens. Self-guidance from the current student policy and external guidance from Qwen3.5-27B share the same inference settings. Both generate with thinking mode enabled, a temperature of 0.7, top-p 0.9, and at most 8,192 new tokens.

C.2 Inference settings

Table 4 summarizes the inference settings shared by the base model and all trained methods for mathematical-reasoning evaluation. We use the fixed random seed `2026` to generate 12 responses per problem, with a sampling temperature of 1.0, top-p 0.95, no top-k restriction, a maximum of 38,912 new tokens, a model context length of 40,960, and thinking enabled. Following the OPSD evaluation protocol (Zhao et al., 2026), we evaluate checkpoints every 20 steps up to 100 steps. Overall Math Avg. is the macro-average of the three mathematical benchmark scores.

Table 3: **Training settings for SFT, GRPO, OPSD, and AIR-OPD.** Lengths are measured in tokens; the SFT limit applies to the full supervised sequence. A dash denotes a setting that is not applicable. For AIR-OPD, one rollout is sampled per dialogue state: the initial response can be followed by at most $K - 1$ retries.

Parameter	SFT	GRPO	OPSD	AIR-OPD
Optimization and adaptation				
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}	5×10^{-6}
Effective batch size	32	32	32	32
Training precision	BF16	BF16	BF16	BF16
LoRA rank / alpha	64 / 128	64 / 128	64 / 128	64 / 128
Optimizer steps	100	500	100	100
SFT maximum sequence length	16,000	–	–	–
On-policy sampling				
Maximum completion length	–	16,000	8,192	8,192
Rollouts per prompt / dialogue state	–	8	1	1
Sampling temperature	–	1.2	1.1	1.1

Table 4: **Inference settings for mathematical-reasoning evaluation.** The base model and all trained methods use these settings on AIME24, AIME25, and HMMT25, without repair guidance.

Parameter	Value
Maximum new tokens	38,912
Maximum model length (tokens)	40,960
Thinking mode	Enabled
Sampling temperature	1.0
Top-p	0.95
Top-k	–
Samples per problem	12
Random seed	`2026`

Out-of-distribution evaluation. MMLU-Pro is evaluated on a fixed 1,000-question subset that covers all 14 subject categories and is released with our code, and GPQA on the 198-question GPQA-Diamond split. For the base model and every trained method, both benchmarks draw four samples per question (Avg@4) with a sampling temperature of 1.0, top-p 0.95, no top-k restriction, and the random seed `2026071201`. MMLU-Pro applies the chat template with thinking enabled, a maximum of 8,192 new tokens, and a model context length of 12,288. GPQA feeds the question prompt to the model without the chat template, with a maximum of 38,912 new tokens and a model context length of 40,960. The predicted option is read from the last `\boxed{}` in the response, and MMLU-Pro additionally accepts an explicit final-answer statement.

D Additional Results and Examples

D.1 Worked repair examples

We reconstruct two complete repair trajectories from the training-time repair logs of the Qwen3-4B self-guidance run, both repaired at the second round. In the first the retry fixes the anchored slip but returns the identical wrong answer, and in the second the retry adopts the corrected equation yet slips again in the very next step although the guidance is valid. The two trajectories illustrate the rationale of the outcome-aware stage weighting in Section 4.3. Because a retry can fail even under valid guidance, a failed retry is not penalized, and the weighting grants its outcome bonus only to the stage whose guidance leads to the verified repair. In each round box, the error anchor a_t located by the guidance generator is highlighted in the failed response, and the supervised repair region R_t opens at the anchor and ends at the endpoint Eq. 7 selects for that stage. Elisions are marked [. . .].

Example 1: repaired in two rounds. The initial attempt counts $\binom{20}{2} = 190$ choices for Terry and $\binom{18}{2} = 153$ for Mary but writes the product as 29,170, counts only the two-red and two-blue cases, and reduces the resulting fraction with a faulty remainder chain to $12/139$, answering 151. The first-round guidance anchors the earliest error, the multiplication slip; the retry corrects the product to 29,070 yet repeats the earlier remainder chain during the reduction and returns the identical 151, so $s_0 = 0$. The second-round guidance targets the remaining omission, the mixed one-red-one-blue case, and the following retry sums all three cases and reaches the verified 441, so $s_1 = 1$.

Example 1, round 0 — failed attempt y_0 (verified incorrect)

Problem. A jar has 10 red candies and 10 blue candies. Terry picks two candies at random, then Mary picks two of the remaining candies at random. Given that the probability that they get the same color combination, irrespective of order, is m/n, where m and n are relatively prime positive integers, find $m + n$.

[. . .] (Terry has $\binom{20}{2} = 190$ choices and Mary has $\binom{18}{2} = 153$) So the total number of possible pairs of choices (Terry picks first, then Mary) is:

$$190 \times 153 = 29170$$

[. . .] (only two favorable cases are considered, both picking two reds and both picking two blues, each counted as $\binom{10}{2}\binom{8}{2} = 45 \times 28 = 1260$)

Total favorable outcomes:

$$1260 + 1260 = 2520$$

[. . .] (a Euclidean reduction of $2520/29170$ with a faulty remainder chain gives)

$$\frac{2520}{29170} = \frac{12}{139}$$

So $m = 12$, $n = 139$, and:

$$m + n = \boxed{151}$$

Example 1 — generator output after round 0: a_0 and g_0

Error anchor a_0. `190 \times 153 = 29170`

Repair guidance g_0. The incorrect solution miscalculates the total number of possible outcomes as $190 \times 153 = 29170$, but the correct value is $190 \times 153 = 29070$. This error arises from an arithmetic mistake in multiplication. [. . .] The correct logic is to compute $\binom{20}{2} \times \binom{18}{2} = 190 \times 153 = 29070$, ensuring the denominator is accurate for the probability computation.

Example 1, round 1 — retry y_1 (verified incorrect, $s_0 = 0$)

[. . .] (the retry adopts the correction) So the total number of possible outcomes (Terry's pick $\times$ Mary's pick) is:

$$190 \times 153 = \boxed{29070}$$

So the correct total number of outcomes is 29070, not 29170. [. . .] (the same two cases are counted again)

Total favorable outcomes

$$1260 + 1260 = 2520$$

[. . .] (the reduction repeats the earlier faulty remainder chain, $29070 \div 2520 = 11$ remainder 1050, and lands on the same fraction)

$$\frac{2520}{29070} = \frac{2520 \div 210}{29070 \div 210} = \frac{12}{139}$$

Thus,

$$m = 12, \quad n = 139 \Rightarrow m + n = \boxed{151}$$

Example 1 — generator output after round 1: a_1 and g_1

Error anchor a_1. `Total favorable outcomes`

Repair guidance g_1. Add the 8100 favorable outcomes from Case 3 (both picking one red and one blue) to the total, resulting in $1260 + 1260 + 8100 = 10620$

Example 1, round 2 — retry y_2 (verified correct, $s_1 = 1$)

[. . .] (the retry keeps the corrected total 29070, recounts Cases 1 and 2, and adds the new case)

Case 3: Both pick one red and one blue. Terry can pick one red and one blue in $\binom{10}{1} \times \binom{10}{1} = 100$ ways [. . .] after Terry's pick, 9 reds and 9 blues remain, so Mary has $\binom{9}{1} \times \binom{9}{1} = 81$ ways, and the total for this case is $100 \times 81 = 8100$.

Total favorable outcomes:

$$1260 + 1260 + 8100 = \boxed{10620}$$

[. . .] (a fresh Euclidean run gives $\gcd = 90$, and 118 and 323 are checked to be coprime)

$$\frac{10620}{29070} = \frac{10620 \div 90}{29070 \div 90} = \frac{118}{323}$$

Hence:

$$m = 118, \quad n = 323 \Rightarrow m + n = \boxed{441}$$

Example 2: a retry that fails under valid guidance. The initial attempt equates the binomial probabilities of exactly two and exactly three heads but mishandles the $(3/4)$ exponent while cancelling the common factor, reducing the equality to $\binom{n}{2} = \binom{n}{3} \cdot \frac{3}{16}$ and reaching $n = 18$. The first-round guidance anchors this equation and derives the correct coefficient $\frac{1}{3}$. The retry derives the correct equation but mishandles the right-hand side while clearing denominators. The next guidance recommends multiplying by 18 to clear all denominators in one step, after which the retry succeeds.

Example 2, round 0 — failed attempt y_0 (verified incorrect)

Problem. An unfair coin lands on heads with a probability of $\frac{1}{4}$. When tossed n times, the probability of exactly two heads is the same as the probability of exactly three heads. What is the value of n?

[. . .] (the response equates the two binomial probabilities)

$$\binom{n}{2} \left(\tfrac{1}{4}\right)^2 \left(\tfrac{3}{4}\right)^{n-2} = \binom{n}{3} \left(\tfrac{1}{4}\right)^3 \left(\tfrac{3}{4}\right)^{n-3}$$

[. . .] (dividing both sides by the common factor $(\frac{1}{4})^2(\frac{3}{4})^{n-2}$, the equation becomes)

$$\binom{n}{2} = \binom{n}{3} \cdot \frac{1}{4} \cdot \frac{3}{4}$$

So now, simplify:

$$\binom{n}{2} = \binom{n}{3} \cdot \tfrac{3}{16}$$

Now plug them in:

$$\frac{n(n-1)}{2} = \frac{n(n-1)(n-2)}{6} \cdot \frac{3}{16}$$

[. . .] (multiplying by 6 and dividing by $3n(n-1)$ gives)

$$1 = \frac{1}{16}(n-2) \Rightarrow 16 = n - 2 \Rightarrow n = 18$$

Final Answer:

$$\boxed{18}$$

Example 2 — generator output after round 0: a_0 and g_0

Error anchor a_0. `binom(n,2) = binom(n,3) * \frac{3}{16}`

Repair guidance g_0. The incorrect solution incorrectly simplifies the exponent terms after dividing by the common factors. The correct coefficient after simplification should be $\frac{1}{3}$, not $\frac{3}{16}$. This arises from miscalculating the exponents: the right-hand side should be $\binom{n}{3} \cdot \frac{1}{4} \cdot \frac{4}{3}$ (from $(1/4)^1 \cdot (3/4)^{-1}$), which simplifies to $\frac{1}{3}$. [. . .] The correct equation after dividing by the common factor is $\binom{n}{2} = \binom{n}{3} \cdot \frac{1}{3}$, not $\frac{3}{16}$.

Example 2, round 1 — retry y_1 (verified incorrect, $s_0 = 0$)

[. . .] (the retry rederives the cancellation with the correct exponents)

$$\binom{n}{2} = \binom{n}{3} \cdot \left(\tfrac{1}{4}\right)^{3-2} \cdot \left(\tfrac{3}{4}\right)^{(n-3)-(n-2)} = \binom{n}{3} \cdot \frac{1}{4} \cdot \frac{4}{3} = \binom{n}{3} \cdot \frac{1}{3}$$

This is the correct simplified equation. [. . .]

$$\frac{n(n-1)}{2} = \frac{n(n-1)(n-2)}{6} \cdot \frac{1}{3}$$

Multiply both sides by 6 to eliminate denominators:

$$3n(n-1) = \frac{1}{3}n(n-1)(n-2) \cdot 6 \Rightarrow 3n(n-1) = 2n(n-1)(n-2)$$

Divide both sides by $n(n-1)$ [. . .]

$$3 = 2(n-2) \Rightarrow 3 = 2n - 4 \Rightarrow 2n = 7 \Rightarrow n = \boxed{7}$$

Final Answer:

$$\boxed{7}$$

Example 2 — generator output after round 1: a_1 and g_1

Error anchor a_1. `Multiply both sides by 6 to eliminate denominators`

Repair guidance g_1. Correct the multiplication factor to 18 instead of 6 when eliminating denominators in the equation

Example 2, round 2 — retry y_2 (verified correct, $s_1 = 1$)

[. . .] (the retry again reaches $\binom{n}{2} = \frac{1}{3}\binom{n}{3}$ and substitutes the binomial expressions)

$$\frac{n(n-1)}{2} = \frac{1}{3} \cdot \frac{n(n-1)(n-2)}{6} = \frac{n(n-1)(n-2)}{18}$$

Now eliminate $n(n-1)$ from both sides [. . .]

$$\frac{1}{2} = \frac{n-2}{18}$$

Multiply both sides by 18:

$$9 = n - 2 \Rightarrow n = \boxed{11}$$

Final Answer:

$$\boxed{11}$$