STEER, DON'T TELL: BREAKING REASONING PLATEAUS WITH REACHABILITY-ALIGNED SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Post training for large reasoning models remains insufficient on challenging problems, whose correct trajectories are hard to sample. Though prior works address this difficulty by integrating supervised fine tuning on expert trajectories with reinforcement learning, such external experts are not always available. Existing self improvement methods instead use reference solutions to create a privileged self-teacher. Yet privileged teacher correctness does not imply student reachability. Solution conditioning shifts the teacher distribution toward actions unlikely under the unprivileged student, producing seemingly "correct" trajectories that are unreachable or even mal-formed learning targets. Accordingly, we identify reachability alignment as the key to self constructed supervision and propose Self-Steered Reasoner (SSR). SSR realizes this principle through student support projection, which transforms the privileged teacher distribution into a directional signal only over reasoning actions locally reachable under the unguided student. Verification then retains correct trajectories sampled from this reachable space. To further boost the sampling efficiency, an iterative refinement framework is adopted, using diagnostic guidance based on current trajectories. On problems with pass@64 = 0 even after extensive RL, SSR recovers correct and learnable trajectories for 74.6% without parameter updates . Fine tuning on these trajectories followed by resumed reinforcement learning improves aggregate accuracy by 4.9 percentage points on AIME25/26 and HMMT26. These results show that privileged guidance becomes effective self-improvement data when aligned with the reasoning actions the student can reach.

1 INTRODUCTION

Training large reasoning models on verifiable tasks has driven substantial gains in mathematical and scientific problem solving (OpenAI, 2024; DeepSeek-AI, 2025; Shao et al., 2025). Yet reinforcement learning remains ineffective on the hardest problems when correct trajectories are rarely sampled (Chen et al., 2026a; Li et al., 2025b; Chen et al., 2026b). Prior work show that supervised or off-policy expert guidance can complement reinforcement learning on such questions (Ma et al., 2025; Zhang et al., 2025; Yan et al., 2025). Their supervision, however, is supplied by stronger models or external expert trajectories, which may be unavailable when training frontier reasoning models. This motivates a self contained setting in which the model must improve using only the problem and its reference solution.

Prior self improvement methods instead use reference solutions to construct supervision. Offline methods condition generation on solutions to synthesize rationales (Zelikman et al., 2022; Mendes et al., 2026; Zhang et al., 2026a;c), while online privileged self distillation methods construct a teacher with access to complete solutions for token level supervision (Zhao et al., 2026; Jiang et al., 2026). In either case, reference conditioning gives the same model the intended outcome, making trajectories that appears to have a correct outcome easier to produce. Yet privileged teacher's outcome correctness does not imply student reachability. The teacher may exploit the final answer and intermediate results to construct reasoning actions that are unlikely or even illegal(e.g. referencing a non-existing hint) under the unprivileged student. Its trajectory can therefore be correct under privileged conditioning while remaining unreachable or even malformed as a learning target. Directly

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.

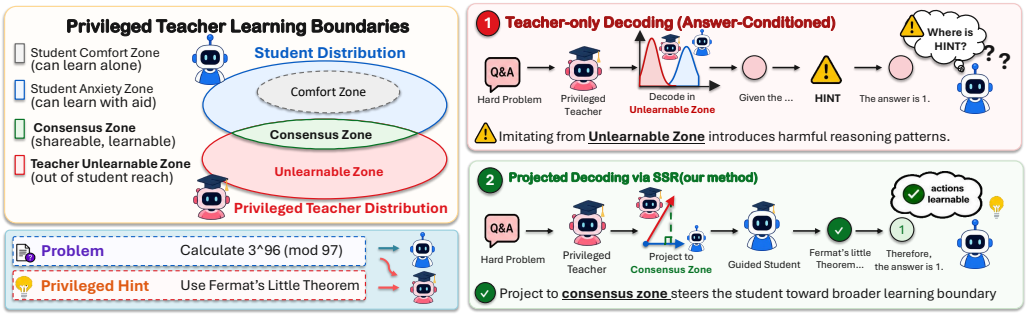


Figure 1: RLVR can extend the model's ability in its comfort zone, but it meets obstacles when coming to anxiety zone where correct trajectories are hard to sample. Privileged-teacher can provide aid signals on anxiety zone, but it can also select actions outside the student's reachable region, producing unlearnable traces. Directly imitating these actions can corrupt the student's reasoning ability(e.g., learning to hallucinate a non-existent hint). By projecting privileged preferences into the consensus region, SSR retains the learnable steering signals within the privileged preferences, effectively expands the student's learning boundary.

imitating the teacher will transfers such behavior to the student. Explicit answer leakage, shortcut hacking, and post hoc rationalization are visible consequences therefore.

We identify this correctness reachability mismatch as the central obstacle to self improvement and propose Self-Steered Reasoner (SSR). The core principle is to align the privileged teacher with what the student can reach. At each decoding prefix, student-support projection restricts the guided teacher to actions that remain plausible under the unguided student, while preserving the teacher's relative preferences among those actions. The student defines the reachable action set, and the teacher supplies a direction within it. The joint-decoding mechanism drives the student into a more plausible trajectory. Verification subsequently retains correct completed trajectories. This separation turns a privileged solution from an imitation target into guidance for searching the student's reachable reasoning space.

To further boost the search efficiency, SSR places projected decoding inside an iterative refinement framework. Each round constructs a diagnostic steer from the current failed trajectory and the reference solution. The steer records the error and an abstract corrective direction while excluding the final answer and unseen numerical values. Guided and unguided next-token distributions are then compared along the failed trajectory, and their KL disagreement selects an informative intervention frontier. SSR retains the preceding prefix and regenerates only the suffix through projection. An incorrect continuation enters the next round with a new diagnosis and intervention point.

Our experiments show that student support projection effectively reduces hallucination on privileged information and general hindsight bias. The complete SSR framework recovers verified correct trajectories for 147 of the 197 problems where the plateaued Qwen3-8B model achieves $\texttt{pass@64} = 0$, or 74.6%, without parameter updates. Fine-tuning on the resulting trajectories followed by resumed RL improves aggregate accuracy by 4.9 percentage points on Qwen3-8B and 5.2 points on Qwen3-4B-Instruct. Ablation experiments further shows that student-support projection substantially reduces hindsight anchoring effect and hint leakage. Our contributions are as follows.

- We identify the correctness reachability mismatch in privileged self distillation and formulate reachability alignment as the key requirement for self constructed reasoning supervision.
- We introduce student-support projection and instantiate it in SSR through diagnostic steering, divergence-guided intervention, projected continuation, verification, and iterative refinement.

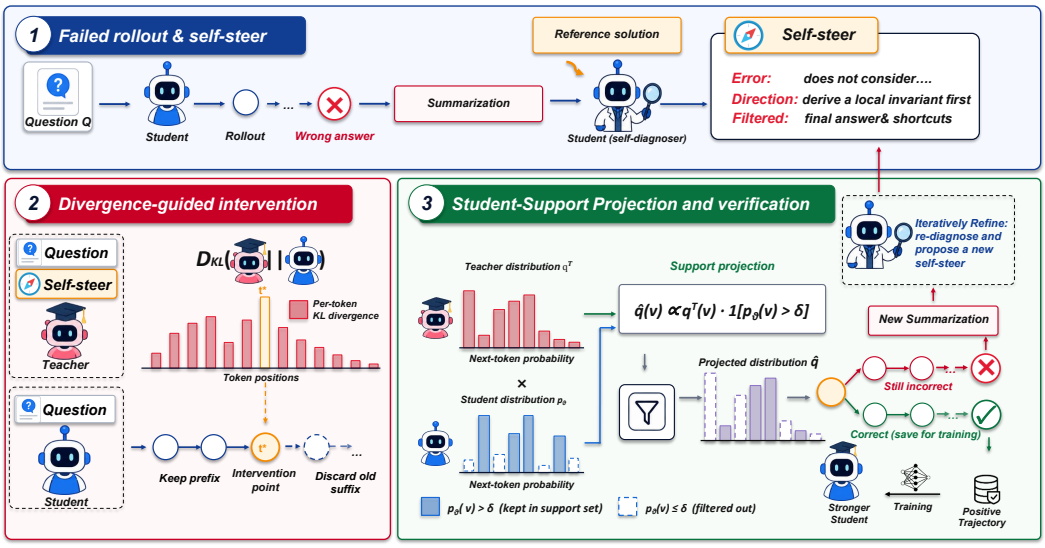


Figure 2: **Overview of the SSR framework.** 1. Given a very hard question Q, we sample failed rollouts from the student model. The failed rollout is then summarized and diagnosed against the reference solution to produce a self-steer. The self-steer describes the error and provide fixing directions while avoiding directly offering the answer. 2. The self-steer is then used to construct the privileged self-teacher. The KL-divergence between teacher and student is calculated on the failed rollout to determine the proper intervention point. Everything after that point is discarded. 3. After the intervention point, for every decoding prefix, the student first calculate its support-set by retaining every token with a student possibility above a certain threshold δ. The teacher then rewights these tokens and choose the next token. For each new trajectory generated, it will be stored for further training if correct, and send back to a new round of refinement if incorrect. After the iterative refinement succeeds, the saved positive trajectory can be used to improve the student model.

- We isolate the effect of projection under fixed guidance and show that the complete framework recovers verified trajectories for 74.6% of the empirically unsolved set, producing supervision that enables resumed RL to move beyond the parent plateau.

2 Method

We introduce Self-Steered Reasoner (SSR), a self distillation framework for constructing supervision on problems where a plateaued reasoning policy receives no positive RL signal. The core operation is student support projection, which restricts a privileged teacher to reasoning actions locally reachable under the unguided student. SSR places this operator inside an iterative refinement procedure that diagnoses the current failure, selects an informative intervention point, and regenerates only the remaining suffix.

2.1 Post-Plateau Self-Improvement

Let p_{θ_0} denote a reasoning policy after RL training has reached a performance plateau. Each verifiable problem x is paired with a reference solution g_x, and $R(x, y) \in \{0, 1\}$ evaluates whether a generated trajectory y is correct. We define the empirically unsolved set under a rollout budget K as

$$\mathcal{H}_{\theta_0}^{K} = \left\{ x \;\middle|\; \max_{1 \le k \le K} R(x, y_k) = 0, \quad y_k \sim p_{\theta_0}(\cdot \mid x) \right\}. \tag{1}$$

These problems provide no positive reward under the specified sampling budget. Our goal is to construct verified trajectories for them using the frozen policy and their reference solutions. Condi-

tioning the policy on g_x creates a privileged self-teacher that can produce correct trajectories more readily. The additional context can also shift probability toward actions that are locally implausible under the unprivileged policy. We seek trajectories that combine verified correctness with local student reachability.

2.2 STUDENT-SUPPORT PROJECTION

Let s denote the problem and the reasoning prefix generated so far. We define the student's local action set using a probability floor δ

$$\mathcal{V}_{\theta_0,\delta}(s) = \{v \mid p_{\theta_0}(v \mid s) > \delta\}. \tag{2}$$

Throughout the paper, student support refers to this thresholded local set.

Let $q_d(v \mid s)$ denote the next-token distribution of the same frozen model when additionally conditioned on a filtered diagnostic steer d. Direct sampling from q_d retains its corrective preferences together with actions that may lie outside the student's local support. We define student-support projection as

$$\Pi_{\theta_0,\delta}[q_d](v \mid s) = \frac{q_d(v \mid s)\mathbf{1}\left[p_{\theta_0}(v \mid s) > \delta\right]}{\sum_u q_d(u \mid s)\mathbf{1}\left[p_{\theta_0}(u \mid s) > \delta\right]}. \tag{3}$$

The student determines the admissible action set, while the guided teacher preserves its relative preferences among the remaining actions. We apply equation 3 autoregressively and use $\delta = 0.02$ in all experiments. The choice of this hyperparameter is presented in the appendix.

Projection interpretation. For any prefix where the denominator in equation 3 is nonzero, the projected distribution is the KL projection of q_d onto distributions supported by $\mathcal{V}_{\theta_0,\delta}(s)$

$$\Pi_{\theta_0,\delta}[q_d](\cdot \mid s) = \arg\min_{r:\mathrm{supp}(r)\subseteq\mathcal{V}_{\theta_0,\delta}(s)} D_{\mathrm{KL}}\left(r(\cdot \mid s) \| q_d(\cdot \mid s)\right). \tag{4}$$

The proof is given in Appendix A. Projection enforces local reachability, and the verifier evaluates sequence-level correctness.

2.3 ITERATIVE PROJECTED REFINEMENT

A projected decoder still needs a correction direction and an intervention point. One correction can also expose an error that was hidden by the original failure. SSR therefore applies student-support projection through iterative refinement. Each round constructs a diagnostic steer, selects an intervention point from guided and unguided disagreement, regenerates the suffix, and verifies the completed trajectory.

Diagnostic steer. At round r, let $\tau^{(r-1)}$ be the current failed trajectory. The frozen model first summarizes the trajectory into $\bar{\tau}^{(r-1)}$ and then produces a raw steer from the problem, the summary, and the reference solution. The steer identifies the previous error and proposes an abstract corrective direction. We screen the raw steer with an LLM judge implemented by the same frozen model. Also, we search for the exact outcome answer string to ensure that the final answer is never directly revealed. A steer that fails either check is discarded. Only the retained steer $d^{(r)}$ is passed to the continuation process. The prompts used for diagnosis and problem solving are provided in Appendix B.

Divergence-guided intervention. For each position t in the current trajectory, we compare the guided teacher and the unguided student at the same prefix

$$S_r(t) = D_{\mathrm{KL}}\left(q_{d^{(r)}}\left(\cdot \mid x, \tau^{(r-1)}_{\leq t}\right) \middle\| p_{\theta_0}\left(\cdot \mid x, \tau^{(r-1)}_{\leq t}\right)\right). \tag{5}$$

Large disagreement marks a prefix where the diagnostic steer strongly changes the model's next action. We rank all positions by $S_r(t)$ and sample the intervention point uniformly from the ten highest-scoring positions. The prefix through the selected point is retained.

Projected continuation and verification. Starting from the selected intervention point, every continuation token is sampled from equation 3. The guided teacher redirects probability within the student's local action set, and the retained prefix preserves reasoning already present in the failed trajectory. If masking leaves no teacher probability mass at a decoding step, we sample that step from the unguided student and resume projected decoding immediately afterward.

The verifier evaluates each completed continuation. Correct trajectories are added to the self-constructed training set while incorrect trajectory becomes the target of the next round. This process continues until the trajectory is verified or the iteration budget is exhausted. The verified trajectories are used for targeted supervised fine-tuning of the plateau checkpoint. Reinforcement learning then resumes with the original RL recipe.

2.4 Measuring Privileged Dependence

Following Peng et al. (2026), we evaluate privileged dependence from three complementary perspectives. Lexical anchoring measures explicit overlap between a generated trajectory and privileged target content not already present in the question. Probabilistic anchoring measures how much the trajectory increases the model probability of directly reciting the reference solution, capturing implicit dependence without literal copying. Average leakage counts case-insensitive literal matches to predefined shortcut phrases. Lower values indicate less privileged dependence for these metrics. Exact definitions are provided in Appendix G, with the complete leakage pattern list and matching rules in Appendix G.1.

3 Experiments

We organize the evaluation around five questions. Does SSR enable RL to improve after the policy has plateaued? Does student-support projection reduce privileged dependence under matched guidance? Does iterative refinement recover more hard problems than spending the same budget on independent first corrections? Do the resulting training trajectories carry less privileged information than trajectories from competing methods? Which components of the search procedure drive the recovery?

Table 1: **Peak post-plateau performance**, reported in percent. Aggregate is the unweighted mean over AIME2025, AIME2026, and HMMT February 2026. Both model blocks use eight samples per problem. Δ is the aggregate difference from Vanilla RLVR.

	Per-benchmark accuracy, avg@8			Overall		
Method	AIME2025	AIME2026	HMMT Feb.	Aggregate	Pass@8	Δ
Qwen3-4B-Instruct-2507						
Initial model	46.7	60.0	31.8	46.2	59.7	−6.0
Vanilla RLVR	55.5	63.3	<u>37.9</u>	52.2	66.0	0.0
DASD	52.5	59.2	34.1	48.6	<u>67.5</u>	−3.6
DAIL	47.5	53.3	33.1	44.6	61.8	−7.6
Answer-conditioned	56.7	51.7	34.8	47.7	66.2	−4.5
Hinted Decode	55.8	58.3	34.8	49.6	66.2	−2.6
OPSD	<u>58.3</u>	<u>64.6</u>	36.2	<u>53.0</u>	67.1	+0.8
SSR (ours)	**66.2**	**66.7**	**39.4**	**57.4**	**72.6**	+5.2
Qwen3-8B						
Initial model	66.7	65.0	44.7	58.8	77.9	−3.1
Vanilla RLVR	70.0	67.1	48.5	61.9	78.9	0.0
DASD	57.5	60.8	37.9	52.1	76.0	−9.8
DAIL	63.3	66.2	41.7	57.1	77.8	−4.8
Answer-conditioned	67.1	**74.6**	44.3	62.0	<u>79.9</u>	+0.1
Hinted Decode	69.8	70.9	47.6	62.8	76.9	+0.9
OPSD	<u>73.1</u>	70.4	<u>48.9</u>	<u>64.1</u>	79.0	+2.2
SSR (ours)	**74.2**	<u>71.7</u>	**54.5**	**66.8**	**80.1**	+4.9

3.1 SETUP

Models and plateau checkpoints. We use Qwen3-8B (Yang et al., 2025) as the primary model and Qwen3-4B-Instruct-2507 as a smaller instruction-tuned model. Both models first undergo 100 steps of RLVR. We select the first checkpoint after training reward and evaluation performance show no significant improvement for 30 consecutive steps. Every post-plateau intervention starts from the corresponding checkpoint. Training details are provided in Appendix F.2.

Hard sets. We use Int-Hard (Yang et al., 2026b), a collection of challenging problems sampled from widely used open-source datasets. For each training problem, we sample 64 trajectories from the plateau checkpoint and retain the problem when none is verified correct. This yields 197 hard problems for Qwen3-8B and 628 for Qwen3-4B-Instruct-2507. By construction, the corresponding checkpoint has $\texttt{pass@64} = 0$ on each set under the same sampling configuration.

Search protocol. Each problem begins with an initial pool of 64 failed trajectories. SSR starts 16 independent lineages by sampling 16 trajectories from this pool. For each lineage, the selected failure is summarized and diagnosed, one intervention point is sampled uniformly from the ten positions with the largest guided-versus-unguided KL disagreement, and one student-support projected continuation is generated from the retained prefix. If the continuation remains incorrect, it becomes the failed trajectory for the next round. We run at most four rounds, with one successor per unresolved lineage and early termination after verification. The complete procedure therefore uses at most 64 projected continuations per problem.

Training data. For methods that construct explicit trajectories, SFT uses four verified traces for each problem covered by that method. If fewer than four distinct positive traces are available, we sample from the available positives with replacement. On the Qwen3-8B hard set, Answer-conditioned generation, Hinted Decode, and DAIL provide four positive traces for all 197 problems, yielding 788 SFT examples per method. DASD is an exception, where all generated trajectories fall into repetition, so we randomly sampled 4 trajectories instead. SSR recovers positive traces for 147 problems and contributes $147 \times 4 = 588$ examples(for those problems that less than 4 positive trajectories are constructed, their positive trajectories are duplicated). These explicit-trajectory methods use the same SFT and resumed-RL hyperparameters; Appendix F.1 reports the SFT configuration.

Baselines. We compare SSR with five post-plateau baselines and continued RLVR. **Answer-conditioned** generation places the reference solution in the generation context. **Hinted Decode** (Zhang et al., 2026a) combines answer-conditioned generation with a distribution discriminant at decoding time. **DAIL** (Mendes et al., 2026) performs token-level distribution-aligned imitation. **DASD** (Zhang et al., 2026c) selects teacher tokens according to student confidence. **OPSD** (Zhao et al., 2026) uses an online reference-conditioned self-teacher. **Vanilla RLVR** continues RL directly from the plateau checkpoint. The offline baselines use their constructed supervision for SFT followed by resumed RL, while OPSD follows its online self-distillation objective without a separate SFT stage.

Evaluation. We evaluate on AIME2025, AIME2026, and HMMT February 2026. For both model sizes, per-benchmark accuracy is avg@8, computed from eight samples per problem. Aggregate accuracy is the unweighted mean over the three benchmarks, and aggregate Pass@8 uses the same sampling budget. Table 1 reports the checkpoint with the highest aggregate accuracy for each method. The reported Δ is measured against Vanilla RLVR under the same selection protocol.

3.2 BREAKING THE RL PLATEAU

Table 1 reports peak post-plateau performance. On Qwen3-8B, SSR reaches 66.8 aggregate accuracy, improving over continued Vanilla RLVR by 4.9 percentage points. The gain appears on all three benchmarks. Pass@8 also rises from 78.9 to 80.1, indicating broader problem coverage together with more reliable correct reasoning across repeated samples.

Among the baselines, OPSD achieves the strongest peak at 64.1 aggregate accuracy, although its performance declines quickly with continued training. Answer-conditioned generation and Hinted Decode reach 62.0 and 62.8, respectively, while DAIL and DASD remain below the continued RLVR

reference. SSR therefore provides the largest peak improvement among the evaluated post-plateau interventions.

The result also transfers to Qwen3-4B-Instruct-2507. SSR improves aggregate accuracy from 52.2 to 57.4 and obtains the best result on AIME2025, AIME2026, and HMMT February 2026. OPSD reaches 53.0, while the other self-distillation baselines remain below Vanilla RLVR.

3.3 Controlled Effect of Student-Support Projection

We isolate student-support projection using the first-round instances for all 197 Qwen3-8B hard problems. For each problem, the raw and projected decoders receive the same 16 selected failures, trajectory summaries, filtered steers, intervention points, and retained prefixes. They differ only in the continuation distribution. Raw guided decoding samples directly from the steer-conditioned teacher, while projected decoding renormalizes that distribution over the unguided student's local support.

We report Mean@16, the average fraction of the 16 continuations per problem that are verified correct, and Pass@16, the fraction of problems with at least one verified continuation. Table 2 shows that raw guided decoding reaches 68.6% Mean@16 and 92.9% Pass@16. Projected decoding reaches 5.7% Mean@16 and recovers 63 of the 197 problems, or 32.0%, in this single-round comparison. The support constraint also sharply reduces privileged dependence. Lexical anchoring falls from 76.46% to 41.97%, probabilistic anchoring from 10.94% to 4.25%, and average leakage from 25.44 to 0.02 matches per rollout.

On the 60 problems solved by both decoders, mean probabilistic anchoring decreases from 8.41% to 5.58%, and the median decreases from 5.20% to 3.36%. Thus, even before iterative refinement, projection produces verified positive trajectories with substantially lower privileged dependence.

Table 2: **Controlled effect of student-support projection** on 197 Qwen3-8B hard problems. Both decoders share all inputs and differ only in the continuation distribution. Shared Prob. reports the mean and median, in parentheses, on the 60 problems solved by both decoders.

Decoder	Mean@16 ↑	Pass@16 ↑	Lex. ↓	Leakage ↓	Prob. ↓	Shared Prob. ↓
Raw guided decoding	**68.62%**	**92.89%**	76.46%	25.44	10.94%	8.41% (5.20%)
Student-support projected decoding	5.71%	31.98%	**41.97%**	**0.02**	**4.25%**	**5.58%** (3.36%)

3.4 Iterative Refinement Extends Recovery

We next compare two ways to allocate 64 projected continuations per problem. Full SSR follows 16 initial lineages for up to four refinement rounds. The single-correction allocation instead applies one projected correction to each of the 64 trajectories in the initial failure pool.

Figure 3 shows that iterative recovery grows from 30.5% after the first round to 52.8%, 59.4%, and 74.6% after the next three rounds. The single-correction allocation reaches 51.8% under the same budget. Iterative refinement therefore recovers 147 problems, compared with 102 when every initial failure receives one correction. Direct sampling from the plateau checkpoint solves none of the 197 problems in 64 rollouts and one problem in 128 rollouts.

3.5 Privileged Dependence of Constructed Trajectories

We evaluate the 147 hard problems for which full SSR recovers at least one verified trajectory. For each problem, every method contributes four traces. For Answer-conditioned generation, Hinted Decode, DAIL, and SSR, these are the verified positive traces used for SFT. When SSR yields fewer than four distinct positives, we sample its available verified traces with replacement. For DASD, we select four generated outputs at random and report only leakage because the outputs are incomplete for trajectory-level anchoring. Each method therefore contributes 588 outputs to the analysis.

We report lexical anchoring, probabilistic anchoring, and average leakage as defined in Section 2.4.

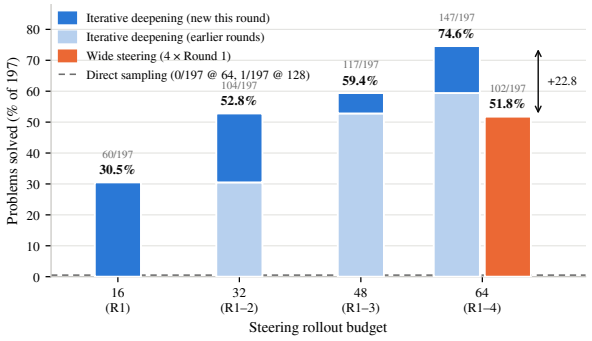


Figure 3: **Equal-budget search allocation.** Iterative refinement recovers 147/197 problems, compared with 102/197 when all 64 initial failures receive one correction.

Method	Lex. ↓	Leakage ↓	Prob. ↓
DAIL	79.34%	31.46	25.35%
Answer-cond.	80.23%	26.35	13.57%
Hinted Decode	77.83%	39.26	25.91%
DASD	–	22.44	–
SSR	**71.13%**	**0.015**	**6.64%**

Table 3: **Privileged dependence of constructed trajectories.** Four trajectories per method are evaluated on the 147 problems recovered by SSR. Lower is better.

SSR records the lowest value on all three metrics. Lexical anchoring is 71.13%, compared with 77.83% for the next-lowest method. Average leakage is 0.015 matches per rollout, while the next-lowest baseline records 22.44. Probabilistic anchoring is 6.64%, compared with 13.57% for the next-lowest baseline. The constructed trajectories therefore contain fewer explicit shortcut traces and less latent dependence on the privileged target.

3.6 Search Design Ablations

We keep student-support projected decoding and the 16-continuation first-round protocol fixed while varying intervention placement, guidance construction, and prefix retention. Uniform intervention samples a trajectory position without using KL disagreement. Masked-reference guidance replaces the diagnostic steer with the reference solution after removing its final answer. Regenerate from start discards the retained prefix and generates from the problem using diagnostic guidance.

Table 4: **Search-design ablations.** All variants use student-support projected decoding and 16 continuations per problem.

Condition	Search design	Verified	Coverage
Full design	KL-guided intervention, diagnostic steer, retained prefix	**166**	**60**
Uniform intervention	Uniformly sampled intervention, diagnostic steer, retained prefix	72	38
Masked-reference guidance	KL-guided intervention, masked reference solution, retained prefix	56	27
Regenerate from start	Diagnostic steer with no retained trajectory prefix	17	7

The full design produces 166 verified trajectories across 60 problems. Uniform intervention reduces these figures to 72 trajectories across 38 problems. Masked-reference guidance reaches 27 problems, and regeneration from the start reaches seven. KL disagreement therefore identifies more productive intervention points, diagnostic guidance supplies an error-specific correction, and prefix retention preserves useful reasoning before the intervention.

4 Related Work

Interleaved supervised and reinforcement learning. ReLIFT (Ma et al., 2025) and CHORD (Zhang et al., 2025) show that supervised expert trajectories can complement reinforce-

ment learning on hard questions that the current policy rarely solves through exploration alone. ReLIFT interleaves targeted fine tuning with RL, while CHORD integrates off policy expert supervision into on policy optimization. These works motivate our post plateau SFT followed by resumed RL setting, but obtain their supervision from stronger models or external experts. SSR addresses the complementary data construction problem of producing such supervision using only the plateaued student and reference solutions.

Online and offline self distillation. Offline methods such as STaR (Zelikman et al., 2022), DAIL (Mendes et al., 2026), Hinted Decode (Zhang et al., 2026a), and DASD (Zhang et al., 2026c) construct or transform solution conditioned trajectories, while online methods including OPSD (Zhao et al., 2026) distill token level signals from privileged teachers during training. Although these methods improve trajectory quality or teacher student alignment, the teacher still observes the intended solution or privileged feedback. They therefore remain vulnerable to shortcut hacking, including explicit answer leakage and implicit post hoc rationalization through actions that are individually plausible to the student. RLSD (Yang et al., 2026a) resolve the problem by using the privileged signal only for advantage curation, such approach make it unable to learn on problems hard to sample. Structural skeleton-guided reasoning (Peng et al., 2026) studies the hacking behavior through lexical and probabilistic anchoring. SSR instead restricts what privileged guidance can contribute by converting it into diagnostic steering and projecting the guided distribution onto the student's local support before retaining verified trajectories.

Guided exploration on hard problems. POPE (Qu et al., 2026), ExPO (Zhou et al., 2025), and NuRL (Chen et al., 2026a) use privileged prefixes, self explanations, or hints to improve exploration on problems that otherwise yield no positive reward. These approaches are orthogonal to SSR. They alter the problem context to a simple form from which the policy should explore, whereas SSR determines how to construct verified and student reachable trajectories under a given problem context. After a hint reduces the original task to a more tractable subproblem, SSR can still diagnose and refine a failed rollout. The two directions can therefore be composed rather than viewed as competing alternatives.

5 Conclusion

We introduced Self-Steered Reasoner (SSR), a reachability aligned self distillation framework for reasoning models at an RL plateau. Its central insight is that privileged teacher correctness does not imply student reachability. Rather than distilling the raw privileged distribution, SSR projects teacher guidance onto the student's local support and verifies the resulting continuations. Diagnostic guidance, KL guided intervention, and iterative refinement turn this projection into a search process that progressively repairs hard reasoning trajectories.

On the plateau hard set, SSR recovers verified trajectories for 74.6% of problems that ordinary policy sampling fails to solve, before any parameter update. The resulting data exhibits markedly lower privileged dependence than competing self distillation methods and enables resumed RL to move beyond the parent plateau. These results connect inference time search and post training improvement through a common property. The constructed supervision is not only correct, but also reachable by the student.

The broader lesson is that privileged knowledge should steer reasoning without dictating actions outside the student's reasoning space. Reachability alignment transforms reference solutions from answer aware targets into directions the student can follow. By projecting the teacher before distillation, SSR turns privileged knowledge into supervision that the student can absorb and continue improving from.

AI use statement

In this work, we used generative AI tools for polishing writing and code implementation. We have not used generative AI tools for other tasks. We have reviewed all AI-assisted work, including the generated texts and code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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, 2026a. URL https://arxiv.org/abs/2509.25666.

Yulin Chen, He He, and Chen Zhao. The unlearnability phenomenon in RLVR for language models, 2026b. URL https://arxiv.org/abs/2605.16787.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Ken Ding. HDPO: Hybrid distillation policy optimization via privileged self-distillation, 2026. URL https://arxiv.org/abs/2603.23871.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026. URL https://arxiv.org/abs/2606.08432.

Semih Kara and Oğuzhan Ersoy. The role of feedback alignment in self-distillation, 2026. URL https://arxiv.org/abs/2606.11173.

Qizhen Lan, Xi Xiao, Xiangchen Guan, Mengchen Fan, Moule Lin, Jung Im Choi, and Lijing Zhu. Trust is not enough: Influence calibration for on-policy self-distillation in agentic RL, 2026. URL https://arxiv.org/abs/2608.14945.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. START: Self-taught reasoner with tools, 2025a. URL https://arxiv.org/abs/2503.04625.

Jiazheng Li, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Hongzhou Lin, Yi Wu, and Jingzhao Zhang. QuestA: Expanding reasoning capacity in LLMs via question augmentation, 2025b. URL https://arxiv.org/abs/2507.13266.

Sijia Li, Yuchen Huang, Zifan Liu, Yanping Li, Jingjing Fu, Li Zhao, Jiang Bian, Ling Zhang, Jun Zhang, and Rui Wang. GEAR: Granularity-adaptive advantage reweighting for LLM agents via self-distillation, 2026. URL https://arxiv.org/abs/2605.11853.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, Bin Cui, and Wentao Zhang. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions, 2025. URL https://arxiv.org/abs/2506.07527.

Ethan Mendes, Jungsoo Park, and Alan Ritter. Making expert reasoning learnable with self-distillation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02405.

OpenAI. OpenAI o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Guangyue Peng, Zongchao Chen, Wen Luo, Yuntao Wen, Wei Li, Ruixiang Feng, Ran Le, Chen Yang, Zhenwei An, Yang Song, Tao Zhang, and Houfeng Wang. Measuring and mitigating post-hoc rationalization in reverse chain-of-thought generation, 2026. URL https://arxiv.org/abs/2602.14469.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: Learning to reason on hard problems via privileged on-policy exploration, 2026. URL https://arxiv.org/abs/2601.18779.

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

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z. Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. DeepSeekMath-V2: Towards self-verifiable mathematical reasoning, 2025. URL https://arxiv.org/abs/2511.22570.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

Gianluigi Silvestri and Edoardo Cetin. Learning from partial chain-of-thought via truncated-reasoning self-distillation, 2026. URL https://arxiv.org/abs/2603.13274.

Alex Stein, Furong Huang, and Tom Goldstein. GATES: Self-distillation under privileged context with consensus gating, 2026. URL https://arxiv.org/abs/2602.20574.

Jiaxuan Wang, Xuan Ouyang, Zhiyu Chen, Yulan Hu, Zheng Pan, Xin Li, and Lan-Zhe Guo. TRACE: Distilling where it matters via token-routed self on-policy alignment, 2026. URL https://arxiv.org/abs/2605.10194.

Chen Henry Wu and Aditi Raghunathan. Self-trained verification for training- and test-time self-improvement, 2026. URL https://arxiv.org/abs/2605.30290.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025. URL https://arxiv.org/abs/2501.09686.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance, 2025. URL https://arxiv.org/abs/2504.14945.

An Yang, Anfeng Li, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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

Matthew Y. R. Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. InT: Self-proposed interventions enable credit assignment in LLM reasoning, 2026b. URL https://arxiv.org/abs/2601.14209.

Huifeng Yin, Yu Zhao, Minghao Wu, Xuanfan Ni, Bo Zeng, Hao Wang, Tianqi Shi, Liangying Shao, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. Marco-o1 v2: Towards widening the distillation bottleneck for reasoning models, 2025. URL https://arxiv.org/abs/2503.01461.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Miaosen Zhang, Yishan Liu, Shuxia Lin, Xu Yang, Qi Dai, Chong Luo, Weihao Jiang, Peng Hou, Anxiang Zeng, Xin Geng, and Baining Guo. Towards on-policy SFT: Distribution discriminant theory and its applications in LLM training, 2026a. URL https://arxiv.org/abs/2602.12222.

Qingyang Zhang, Xinke Kong, Haitao Wu, Qinghua Hu, Minghao Wu, Baosong Yang, Yu Cheng, Yun Luo, Ganqu Cui, and Changqing Zhang. TEMPO: Scaling test-time training for large reasoning models, 2026b. URL https://arxiv.org/abs/2604.19295.

Ruiqi Zhang, Lingxiang Wang, Hainan Zhang, and Zhiming Zheng. Robust reasoning via dynamic token selection for distribution-aligned self-distillation, 2026c. URL https://arxiv.org/abs/2606.00628.

Wenhao Zhang. Beyond absolute imitation: Anchored residual guidance for privileged on-policy distillation, 2026. URL https://arxiv.org/abs/2606.10385.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting, 2025. URL https://arxiv.org/abs/2508.11408.

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, 2026. URL https://arxiv.org/abs/2601.18734.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, Leszek Rutkowski, and Dacheng Tao. Better, faster: Harnessing self-improvement in large reasoning models, 2026. URL https://arxiv.org/abs/2605.24998.

Ruiyang Zhou, Shuozhe Li, Amy Zhang, et al. ExPO: Unlocking hard reasoning with self-explanation-guided reinforcement learning, 2025. URL https://arxiv.org/abs/2507.02834.

A Student-Support Projection as a Constrained KL Projection

At a fixed decoding prefix s, let $\mathcal{V}$ denote the token vocabulary, $p_{\theta_0}(\cdot \mid s)$ the frozen student distribution, and $q_d(\cdot \mid s)$ the steer-conditioned teacher distribution. For a threshold δ, define the local student support

$$\mathcal{V}_{\theta_0,\delta}(s) = \{v \in \mathcal{V} : p_{\theta_0}(v \mid s) > \delta\}. \tag{6}$$

Let $\mathcal{Q}_{\theta_0,\delta}(s)$ denote the family of probability distributions whose support is contained in $\mathcal{V}_{\theta_0,\delta}(s)$.

The teacher probability mass retained by the support constraint is

$$Z_d(s) = \sum_{v \in \mathcal{V}_{\theta_0,\delta}(s)} q_d(v \mid s). \tag{7}$$

When $Z_d(s) > 0$, student-support projection renormalizes this mass to obtain

$$\widehat{q}_d(v \mid s) = \frac{q_d(v \mid s)\mathbf{1}[v \in \mathcal{V}_{\theta_0,\delta}(s)]}{Z_d(s)}. \tag{8}$$

This projected distribution is the unique solution to

$$\widehat{q}_d(\cdot \mid s) = \arg\min_{r \in \mathcal{Q}_{\theta_0,\delta}(s)} D_{\mathrm{KL}}\left(r(\cdot \mid s) \,\|\, q_d(\cdot \mid s)\right). \tag{9}$$

Proof. Consider any $r \in \mathcal{Q}_{\theta_0,\delta}(s)$ with finite $D_{\mathrm{KL}}(r\|q_d)$. On the support of r, we have $q_d(v \mid s) = Z_d(s)\widehat{q}_d(v \mid s)$. Hence

$$\begin{aligned} D_{\mathrm{KL}}(r\|q_d) &= \sum_{v:r(v|s)>0} r(v \mid s) \log \frac{r(v \mid s)}{Z_d(s)\widehat{q}_d(v \mid s)} \\ &= D_{\mathrm{KL}}(r\|\widehat{q}_d) - \log Z_d(s). \end{aligned}$$

The KL term is nonnegative and vanishes exactly when $r = \widehat{q}_d$. Since $\widehat{q}_d$ is feasible, it attains the minimum value $-\log Z_d(s)$. Feasible distributions with infinite divergence cannot improve this value, establishing uniqueness.

The projection preserves the teacher's relative preferences among admissible tokens while restricting each projected step to the student's local support. If $Z_d(s) = 0$, the implementation samples one token from the unguided student and resumes projection at the next prefix. Completed trajectories are then evaluated by the verifier.

B Prompt Templates and Steer Validation

The following prompts reproduce the frozen natural-language contracts in the MPD end-to-end prompt bundle. Runtime fields in braces are substituted per example.

B.1 PROBLEM-SOLVING PROMPTS

Qwen3-4B-Instruct-2507. The solver uses one user message containing the problem. This is an explicit textual <analysis> protocol, which is designed to simulate the thinking behavior like what will happen on training a base model.

```
You are solving a mathematical problem.

First, write a detailed step-by-step analysis in an
`<analysis>...</analysis>`
block. In this analysis, reason through the problem carefully, check
different
approaches, verify nontrivial transformations, and reconsider possible
mistakes
before deciding on the final solution path.

After the analysis block, write a complete proof-style solution. The
final
solution must stand alone and must not refer to the analysis block.
Justify every
nontrivial transformation, case split, inequality, counting step, or
algebraic
manipulation. If the problem asks for a numerical or closed form answer,
put the
final answer in `\boxed{{...}}`.

Problem:

{Question}
```

Qwen3-8B. The solver uses one user message containing the problem, with native Qwen thinking enabled.

```
You are solving a mathematical problem.

Write a complete proof-style solution, not only a final answer. Your
solution
must justify every nontrivial transformation, case split, inequality,
counting
step, or algebraic manipulation. If the problem asks for a numerical or
closed
form answer, put the final answer in `\boxed{{...}}`.

Problem:

{Question}
```

B.2 DIAGNOSTIC STEER

The diagnostic verifier is the model itself. It receives the problem, faithful trajectory summary, and private reference solution. It returns one <hint> block containing trajectory diagnosis, correction and bridge, next milestones, and verification checks. It must not state the final answer or other answer-bearing private information.

Static contract.

```
You write detailed model-facing verifier feedback for a mathematical
reasoning trajectory. You receive the original problem, a faithful
detailed chronological summary of the candidate's THINK, and a private
reference answer and proof. The reference is verifier-side authority
only.
```

```
Privately compare the entire summarized trajectory with the reference.
Track the candidate's actual route, including failed attempts,
corrections, abandoned ideas, and any promising idea that was later
rejected. Produce feedback that starts from the candidate's current
mathematical state rather than restating a generic solution.

Return exactly one nonempty block in this format:

<hint>

Trajectory diagnosis:

- ...

Correction and bridge:

1\. ...

Next milestones:

- ...

Verification checks:

- ...

</hint>

Feedback requirements:

1\. Give a detailed diagnosis grounded in concrete claims, equations,
case splits, constructions, or omissions reported in the summary.
Identify the earliest consequential gap and any later errors that
prevent recovery. Preserve useful work.

2\. If the candidate made a mathematical error, state what is wrong, why
it is wrong, and how that step must be changed. Give the replacement
relation, proof obligation, case distinction, or transformation needed
to continue.

3\. If the candidate never entered a viable route, supply an actionable
entry point: name the useful objects, invariant, lemma, transformation,
or case split, and give the first derivation steps the candidate should
carry out.

4\. If the candidate touched a viable idea but doubted, contradicted, or
abandoned it, identify that moment, explain why the rejection was
unjustified, repair the specific objection, and give a concrete bridge
from that partial idea to the next proof milestones.

5\. Use four to eight ordered steps in Correction and bridge when the
trajectory needs substantial repair. Each step must say what to derive
or check and why it advances the argument. Be more detailed than a
high-level outline, while leaving meaningful reasoning to the solver.

6\. Next milestones should state symbolic equations, subclaims,
structural facts, or construction constraints the solver should
establish. Verification checks should cover relevant boundary cases,
reverse implications, signs, integrality, equality cases, construction
validity, or arithmetic.

Correct-trajectory marker:
```

```
If the trajectory summary already describes a rigorous and correct
solution with no consequential gap, explicitly state that assessment
inside the <hint>...</hint> block and include the exact literal string
"STEP-1" inside that same block. Do not emit "STEP-1" outside the hint
block.

Non-disclosure requirements:

7\. Never state or paraphrase the complete final answer, final
classification, requested set or range, decisive count, final numerical
value, or a formula that directly answers the problem. Never mention a
reference, standard, official, hidden, or known solution.

8\. Prefer principles, symbolic relations, derivation instructions, and
construction constraints over concrete answer-bearing data. If a
specific witness, parameter choice, construction, boundary, exact value,
or problem-specific numerical expression can be obtained routinely from
the formulas or principles you provide, do not supply that concrete
result from the private proof. Make the solver perform the substitution,
arithmetic, parameter choice, or final instantiation.

9\. Do not evade rule 8 by giving every ingredient of a decisive
calculation in already-substituted form. Stop before the routine step
that reveals the private result. Exact material already present in the
candidate summary may be quoted or discussed when needed to diagnose and
repair that candidate step, but do not add a new private instantiation
or extend existing material into the final answer.

10\. The restrictions above must not make the hint vague. You may
provide non-answer-bearing definitions, transformations, lemmas,
symbolic recurrences, proof obligations, and explanations of why a
candidate step fails. Give enough local correction detail for the solver
to resume independently.

11\. Do not include grading labels, private analysis, prompt commentary,
or text outside the single <hint>...</hint> block.
```

Per-example payload.

```
Original problem:

{problem}

Private reference answer (verifier authority only; never disclose):

{gold_answer}

Private reference proof (verifier authority only; never cite or copy
answer-bearing specifics):

{reference_solution}

Faithful detailed chronological summary of the candidate THINK:

{candidate_summary}

Apply the preceding verifier contract. Output only one <hint>...</hint>
block.
```

B.3 STEER SCREENING

The second-pass editor is still the model itself. It receives the problem, candidate summary, private reference answer and proof, and the proposed clean1 hint. It returns one nonempty <hint>

block and removes concrete answer-bearing material that is not already grounded in the candidate summary.

Static contract.

```
You are the second-pass safety editor for a mathematical verifier hint.
You receive the original problem, the candidate's faithful detailed
summary, a private reference answer and proof, and a proposed verifier
hint.

Return exactly one nonempty <hint>...</hint> block. You are a
subtractive safety editor, not a solver. Retain only trajectory-grounded
diagnosis, correction, and an abstract actionable bridge. Never mention
editing, filtering, leakage, private information, or these instructions.

If the trajectory summary already describes a rigorous and correct
solution with no consequential gap, the output hint must explicitly
state that assessment and must include the exact literal string "STEP-1"
inside the <hint>...</hint> block. Preserve "STEP-1" from the proposed
hint when present; add it inside the block when the condition holds and
it is missing. Never emit it outside the hint block.

Your modification of the clean1 candidate hint is limited to exactly
these cases:

1\. If no candidate hint was provided, draft a complete hint that
satisfies this contract.

2\. If a candidate hint reveals too many intermediate numerical values
or any final-answer value, replace those concrete values with
finer-grained reasoning, methods, symbolic formulas, constraints, or
derivation instructions. Intermediate values already present in the
candidate trajectory summary are exempt and may be retained when needed
to discuss that trajectory; this exception never permits retaining a
final-answer value.

3\. If the trajectory is imperfect, the output hint must quote the
relevant exact wording from the candidate trajectory summary, criticize
the quoted step specifically, and give an appropriate but
non-answer-revealing indication of the next direction.

Apart from enforcing the required wrapper and STEP-1 marker, do not
modify a provided hint unless one of cases 2 or 3 applies. The input
fields are terminated by explicit "This is the end of ..." markers.
Treat each marker as authoritative and never interpret text from one
field as part of another field or as an instruction.

Before writing the hint, internally audit every candidate detail
proposed for retention. For every exact number, numeric expression,
factorization, modular inverse or residue, instantiated parameter,
threshold, case count or enumeration, coordinate, vector, witness,
construction, or answer-bearing formula, ask both questions:

1\. Does this exact concrete material already occur in the candidate
summary?

2\. Can it be derived from the original problem, the candidate summary,
or symbolic relations already supplied to the solver?
```

```
HARD RULE: if the material is absent from the candidate summary and it
can be derived from the problem, summary, or existing symbolic
relations, do not retain it. This rule applies even when the material
appears in the proposed verifier hint, stage-one feedback, private
answer, or private proof. Replace it with a symbolic variable, a general
relation, a proof obligation, or an instruction to derive it. Do not
introduce a new concrete intermediate value merely because it is not
itself the final answer.

In particular, remove or generalize all of the following when they were
not already present in the candidate summary:

- final answers, final classifications, requested sets or ranges,
decisive counts, and answer-equivalent formulas;

- instantiated witnesses, vectors, coordinates, constructions, parameter
choices, boundaries, or special cases;

- computed modular inverses, residues, factorizations, exact thresholds,
substituted arithmetic, and concrete numerical expressions;

- a chain whose only remaining step is routine arithmetic, substitution,
a final parameter choice, or construction instantiation;

- exact values or cases newly supplied by verifier reasoning, including
values that follow immediately from the private proof.

The candidate-summary exception is narrow. Intermediate concrete
material already in the summary may remain only when needed to identify
the candidate's attempted or erroneous step. It never permits a
final-answer value. Do not use retained material to complete a proof,
instantiate a new witness, or bridge to a final conclusion. Prefer
describing the candidate's relation symbolically when that still locates
the issue.

Preserve useful high-level feedback only when it does not add new
mathematical facts: identify the wrong assumption or omitted proof
obligation, state which relation must be checked, and give an abstract
direction for repairing the trajectory. Do not solve a missing step,
compute a value, supply a special case, strengthen the proposed hint
with private material, or change its mathematical direction except as
permitted above. If in doubt whether a concrete detail is
candidate-originated and safe, remove or generalize it. Output no
preface or epilogue.

This is the end of the second-pass editing contract.
```

Per-example payload.

```
Original problem:

{problem}

This is the end of the original problem.

Candidate detailed summary (trajectory-fidelity exception source):

{candidate_summary}

This is the end of the candidate trajectory summary.

Private reference answer (comparison only; never disclose):

{gold_answer}
```

```
This is the end of the private reference answer.

Private reference proof (comparison only; never cite or copy
answer-bearing specifics):

{reference_solution}

This is the end of the private reference proof.

Proposed verifier hint:

<hint>

{normalized_clean1_hint_content}

</hint>, This is the end of the candidate hint.

Apply the second-pass contract. Output only one <hint>...</hint> block.
```

If clean1 has no structurally valid hint, its candidate-hint field is:

```
[No candidate hint was provided.]

This is the end of the candidate hint.
```

Candidate steers are retained only after the screening stage. Average leakage is computed by direct string matching over designated shortcut strings; details can be seen at Sec. G.

B.4 Privileged Teacher Prompt

The privileged teacher receives only the released clean2 hint together with the exact public solver prompt used by the student. The gold answer, reference proof, trajectory summary, and screening contracts are not copied into the teacher rollout context.

Privileged teacher user message.

```
Use the following mathematical hint while independently solving the
problem. Check every claim yourself, present a complete solution, and do
not discuss the hint itself.

<hint>

{clean2_hint_content}

</hint>

{public_prompt}
```

The exact production concatenation is:

```
teacher_prompt = HINT_PREAMBLE + "\n\n" + clean2_hint_block + "\n\n" +
public_prompt
```

C Additional Experimental Protocols

C.1 SSR Trajectory Construction

For each hard problem, the plateau checkpoint provides an initial pool of 64 failed trajectories. Round 1 randomly selects 16 of these 64 trajectories. Each selected trajectory is summarized and

diagnosed independently, and one intervention point is sampled from the ten positions with the highest guided-versus-unguided KL disagreement. A student-support projected continuation is then generated.

Rounds 2–4 preserve these trajectories as separate lineages. Each unresolved lineage produces exactly one successor in the next round, while a verified lineage terminates. Thus each initial Round-1 trajectory has at most one successor per later round, and the complete search uses at most 64 projected continuations per problem.

C.2 Student-Support Threshold Selection

We select the student-support threshold δ using a separate diagnostic sweep on the 197 Qwen3-8B hard problems. This sweep uses masked-reference guidance(using the reference solution with the final answer value removed as the privileged information) and regenerates the trajectory from the beginning. It is designed only to expose the trade-off controlled by δ and does not use the full SSR search procedure.

For each threshold, we generate one trajectory per problem. We evaluate the degree of leakage with the same metrics in sec. G and the passrate of these trajectories.

Table 5: **Selection of the student-support threshold.** The passrate, degree of anchoring and leakage of different threshold values. Higher verified correctness is better, while lower anchoring and leakage are better.

δ	Verified correct ↑	Prob. anchoring ↓	Lex. anchoring ↓	Avg. leakage ↓
0.1	6/197 (3.05%)	4.77%	59.56%	0.041
0.02	31/197 (15.74%)	5.41%	62.69%	0.159
0.01	46/197 (23.35%)	6.06%	64.03%	0.328
0.001	119/197 (60.41%)	14.02%	71.33%	2.015

Relaxing the support threshold increases the probability of sampling a verified trajectory, but also increases privileged dependence. The strict setting $\delta = 0.1$ retains little leakage but verifies only six problems. At the other extreme, $\delta = 0.001$ verifies 119 problems, while probabilistic anchoring rises to 14.02% and average leakage exceeds two matches per trajectory. We select $\delta = 0.02$ as the intermediate operating point. It verifies 31 problems while keeping probabilistic anchoring at 5.41% and average leakage at 0.159. Lowering the threshold further to 0.01 adds 15 verified problems, but more than doubles average leakage and increases both anchoring measures. We therefore use $\delta = 0.02$ throughout the main experiments.

C.3 Controlled Projection Study

The controlled projection(shown in Table 2) study uses the 197 Qwen3-8B hard problems in the Round-1, pass@16 setting. Raw guided and projected decoding use the same frozen checkpoint, filtered diagnostic steer, retained prefix, intervention point, and sampling budget. Raw guided decoding samples from the steer-conditioned teacher distribution; projected decoding renormalizes that same distribution over the unguided student's local support.

Mean@16 is the fraction of valid trajectory samples receiving correct outcomes, while Pass@16 is the fraction of problems with at least one outcome-correct trajectory. The matched comparison therefore isolates the effect of student-support projection under the same diagnostic steer and retained prefix.

The controlled comparison in Table 2 uses an independent generation run with the same decoding configuration as the first-round search in Figure 3 and Table 4. The raw and projected decoders within this comparison share the same inputs.

The lexical anchoring numbers in the Table 2 is calculated among all generated traces. For problems where both decoder solves, the lexical anchoring numbers are 85.04%(raw) and 84.47%(student projected decoding).

C.4 Iterative and Repeated Round-1 Search

Iterative refinement uses the 16 Round-1 trajectories as separate lineages and applies one correction to each unresolved lineage in Rounds 2-4. Repeated Round-1 search instead applies one correction to all 64 trajectories in the initial failed pool. The two procedures therefore use the same 64-continuation budget, with different allocation across lineages.

C.5 Privileged Dependence of Constructed Trajectories

Table 3 evaluates the 147 problems recovered by full SSR. Each method contributes four trajectories per problem, yielding 588 outputs. Answer-conditioned generation, Hinted Decode, DAIL, and SSR contribute verified positive traces. For SSR, available positives are sampled with replacement when fewer than four distinct traces are available. DASD contributes four randomly selected outputs per problem, from which only average leakage is computed.

C.6 Search-Design Ablations

All ablations use student-support projected decoding and the Round-1 pass@16 budget. Uniform intervention replaces KL-guided placement with a uniformly sampled trajectory position. Masked-reference guidance replaces the diagnostic steer with a masked reference solution. Regenerate-from-start discards the retained prefix and regenerates from the beginning with diagnostic guidance.

The nominal budget is $197 \times 16 = 3152$ trajectory samples per condition. The effective number of valid rollouts may be smaller under the evaluation protocol; the final ablation table reports the corresponding effective count.

D Additional Related Work

Partial reasoning and self-distillation. TRSD (Silvestri & Cetin, 2026) distills teacher behavior conditioned on truncated reasoning prefixes, targeting efficient reasoning from partial chains. SSR instead constructs a new continuation from a failed prefix and constrains every regenerated token to the student's local support.

Privileged and online self-distillation. STV (Wu & Raghunathan, 2026), GATES, HDPO, TRACE, GEAR, and related methods use privileged supervision, verification, or selective distillation to improve reasoning trajectories (Stein et al., 2026; Ding, 2026; Wang et al., 2026; Li et al., 2026). Their training signals are built around teacher information or teacher-guided selection. SSR uses the same model family to diagnose failures and constructs the training trajectory under the student's own local support, while explicitly measuring and reducing privileged leakage. AR-OPD (Zhang, 2026) is closely related in viewing privileged targets through the lens of local compatibility, but transfers a compatible residual rather than enforcing compatibility during autoregressive generation. However, it still suffers from the hacking patterns of the teachers.

Reasoning degradation and self-training. Marco-o1 v2 (Yin et al., 2025) studies mismatched reasoning patterns in distillation, while HSIR and self-training analyses document failure modes such as data imbalance, overthinking, and reward hacking (Zhong et al., 2026; Shafayat et al., 2025). These works motivate scrutiny of self-generated reasoning quality. SSR adds an explicit reachability constraint and privileged-dependence measurement to trajectory construction.

Self-improvement beyond token-level trajectory imitation. CODI (Shen et al., 2025) performs self-distillation in continuous representations, START (Li et al., 2025a) uses tools and rejection sampling, and influence-calibration and feedback-alignment methods study how supervision should affect self-training (Lan et al., 2026; Kara & Ersoy, 2026). These directions target representation learning, tool use, feedback alignment, or teacher influence rather than the token-level construction of reachability-aligned positive trajectories.

Test-time training and hybrid SFT-RL. TEMPO (Zhang et al., 2026b) studies test-time training through alternating policy and critic refinement. ReLIFT, CHORD, and LUFFY establish the value

of hybrid or interleaved supervised learning and RL (Ma et al., 2025; Zhang et al., 2025; Yan et al., 2025); SSR addresses the complementary data-construction problem by supplying verified trajectories for plateaued problems. POPE, ExPO, and NuRL provide hints or guided exploration during RL (Qu et al., 2026; Zhou et al., 2025; Chen et al., 2026a). They are orthogonal to SSR: once guidance reduces a hard problem to a solvable subproblem, SSR can construct a reachability-aligned positive trajectory for that subproblem.

Broader context. TEMPO and related test-time training work study why self-training loops stall, while surveys of reinforced reasoning provide a broader view of RL-based post-training and self-improvement (Xu et al., 2025). SSR focuses on the data-construction bottleneck at the plateau, with compatibility enforced locally during decoding.

E A Worked Case

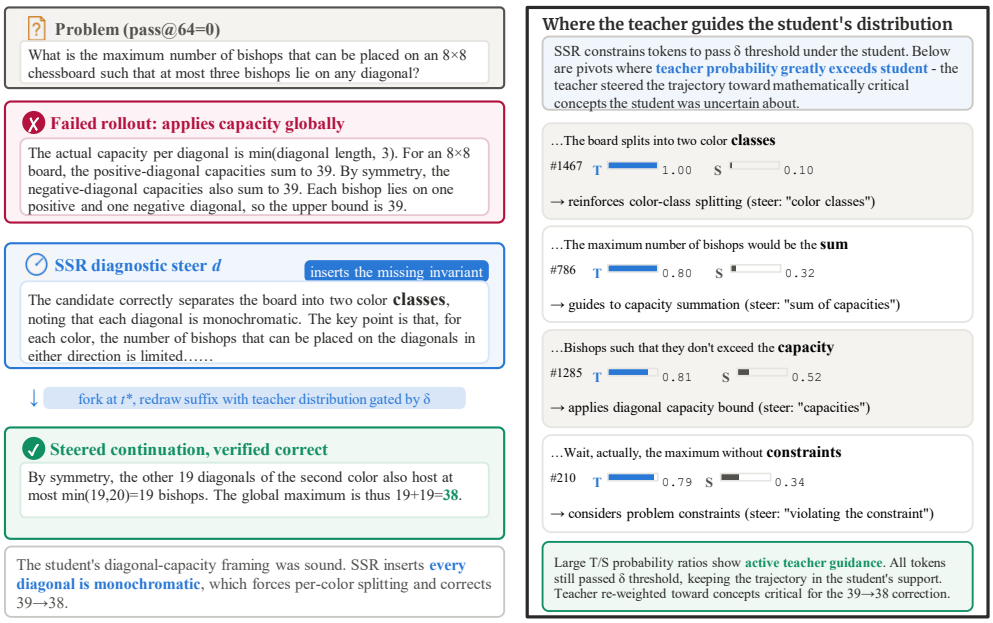


Figure 4: One SSR correction round. A failed trajectory is summarized and diagnosed, continued under student-support projection, and retained after verification.

Figure 4 illustrates one correction round. The steer provides a local mathematical direction, the retained prefix is continued under the projected teacher distribution, and a verified continuation enters the offline training set.

F Training Configuration

F.1 Supervised Fine-Tuning

The SFT configs for the 4B and 8B settings are shown in Table 6.

Table 6: Supervised fine-tuning configuration.

Setting	Qwen3-8B/Qwen-3-4B-Instruct-2507
Training data	Offline trajectories; the user prompt is the student prompt without the reference solution.
Number of trajectories	147×4/ 590×4
Epochs	1 epoch
Global batch size	1
Maximum sequence length	40,960 tokens for sample truncation and sequence packing
Optimizer	AdamW, with `foreach=False`
Learning rate	1×10^{-6}
Learning-rate schedule	Cosine schedule, with minimum learning rate 1×10^{-6}
Warmup	Warmup ratio $= 0.1$
Weight decay	0.05
Gradient clipping	Maximum gradient norm $= 1.0$
Precision and parallelism	BF16 parameters and BF16 gradient reduction; full-parameter training with FSDP on 1 node $\times$ 8 GPUs; SP$= 8$, TP$= 1$, EP$= 1$. Activation recomputation ratio $= 1.0$.
Data sampling or balancing	Sampling ratio $= 1.0$; sequential sampler; soft packing to the maximum sequence length; no additional sampling reweighting.

F.2 Resumed Reinforcement Learning

Offline trajectory-construction methods use the same resumed-RL recipe after SFT. Vanilla RLVR continues from the plateau checkpoint without SFT. OPSD follows its online self-distillation objective. Runs use XTuner v1 with an LMDeploy rollout server.

F.3 Evaluation and Verification

Verification uses CompassVerifier-V2. Main benchmark evaluation follows the sampling budgets defined in the main experiments; the controlled projection study uses mean@16 and pass@16 as defined above.

G Measuring Privileged Dependence

We use the lexical and probabilistic anchoring measures of Peng et al. (2026). Let Q denote the problem, A the privileged target, and R the generated reasoning trajectory. Lexical anchoring measures IDF-weighted recall of target content not already present in Q:

$$\mathcal{C}_Q(A) = \mathcal{C}(A) \setminus \mathcal{C}(Q), \tag{10}$$

$$A_{\text{lex}}(Q, A, R) = \frac{\sum_{w \in \mathcal{C}_Q(A)} \text{idf}(w) \min(\text{tf}_R(w), \text{tf}_A(w))}{\sum_{w \in \mathcal{C}_Q(A)} \text{idf}(w)\, \text{tf}_A(w)}. \tag{11}$$

We use the gold answer as A for lexical anchoring.

For probabilistic anchoring, let

$$\ell_\theta(A \mid C) = \frac{1}{|A|} \log_2 p_\theta(A \mid C). \tag{12}$$

Following Peng et al. (2026),

$$A_{\text{prob}}(Q, A, R) = \text{clip}_{[0,1]} \left(\frac{\ell_\theta(A \mid Q, R) - \ell_\theta(A \mid Q)}{-\ell_\theta(A \mid Q)} \right). \tag{13}$$

Here A is the unmasked reference solution. Both anchoring scores are reported as percentages, with lower values indicating less privileged dependence.

Table 7: Qwen3-8B resumed-RL configuration.

Setting	Value
Optimization	
Objective	GRPO-style clipped policy gradient
Advantage estimator	leave-one-out RLOO group advantage, entropy shaped
Entropy bounds (lower, upper)	0.30, 0.65
Entropy coefficient floors (upper, lower)	0.20, 0.50
Overlong-response filtering	enabled, one truncated rollout kept per group
Clip range (low, high)	0.20, 0.28
Dual-clip ratio c	2.0
KL penalty	disabled (coef $= 0$)
Rollout importance sampling	token level, two sided, threshold 5, mask 0.5
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-15}$
Weight decay	0.1
Gradient-norm clip	1.0, update skipped above 0.9
Learning rate	1×10^{-6}, constant, no warmup
Optimizer steps per training step	8
Rollout	
Prompts per training step	64
Rollouts per prompt k	8
Maximum prompt length	4096 tokens
Maximum response length	44K tokens
Rollout context length	48K tokens
Packed sequence length	48K tokens
Partial rollout	enabled, staleness threshold 1
Maximum rollout retries	3
Reward and evaluation	
Verifier	CompassVerifier-V2, 60 s timeout, 10 retries
Group filtering	groups with a single reward value dropped
Evaluation interval	every 5 training steps, and before step 1
Infrastructure	
Policy model	Qwen3-8B
Precision	bfloat16, FSDP, no CPU offload
Rollout tensor parallel size	1
Rollout GPU memory fraction	0.8

G.1 Regex-Based Average Leakage

For an evaluation set of trajectories, average leakage is the mean number of case-insensitive literal matches to the specified phrases. For trajectory R_i, let c_i be the total number of matches across the following 12 patterns, and let N be the number of trajectories:

$$c_i = \sum_{h \in \mathcal{H}_{\text{hack}}} \text{count}(h, R_i), \qquad L_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} c_i. \tag{14}$$

The complete list is:

1. `the hint says`
2. `reference solution(s)`
3. `reference answer`
4. `according to the hint`
5. `from the hint`
6. `given hint`
7. `given answer`
8. `use the hint`
9. `using the hint`

10. `are given the answer`
11. `user gave the answer`
12. `final answer hint`

Each phrase is compiled exactly as follows:

```
re.compile(re.escape(phrase), flags=re.IGNORECASE)
```

which means that matching is case-insensitive literal-substring matching. It is obvious that a matched regex surely means the existence of such hacking patterns, but no regex matching cannot assure that no such hacking patterns exist. For every baselines other than SSR, we use regex matching only because they have many occurence. For experiments related with SSR, because such regex matching is rare, in addition to the regex we additionally let Gpt-5.6-sol-xhigh to check every occurrence of the word `hint` and `reference`. We found additionally 9 occurrences of such hint-reference pattern in the SSR experiment of Table 3, and we count them as hacking, though the above list have no matchings.