WHERE TO GO, WHAT TO AVOID: STATE-RELATIVE SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing on-policy self-distillation learns from the student's own rollouts, but its teacher signal remains conditioned on a fixed reference solution. This can entangle useful local reasoning with answer-dependent confidence and preference for a particular solution path. We argue that supervision should instead reflect the student's current rollout state, encouraging viable continuations while discouraging directions that have already led to failure. We therefore propose **State-Relative Self-Distillation (SRSD)**, which incorporates state-relative rollout information into supervision through two complementary teacher roles. A State Tracker constructs a state representation from the student's complete on-policy rollout, providing a positive *Where to go* signal for subsequent reasoning. A State Guard leverages a failed trajectory sampled from the same policy to provide a negative *What to avoid* signal, discouraging the student from revisiting failure-inducing directions. Together, these signals encourage valid alternative reasoning paths while reducing unsupported confidence and rigid path preference. We evaluate SRSD on mathematical reasoning, code generation, and agentic tool use across multiple model scales and benchmarks. SRSD consistently delivers strong performance improvements across these domains, with the largest gains in code generation, where it outperforms the strongest compared method by 8 points. Code will be released upon publication to facilitate reproducibility.

1 INTRODUCTION

Training language models to reason well often relies on teacher-generated solutions. However, during generation, the student can reach states that offline solutions do not cover, creating a mismatch between training and inference. On-policy distillation (OPD) reduces this mismatch by letting the student sample its own rollouts while a teacher provides dense supervision on the states the student actually visits (Lu & Lab, 2025; Agarwal et al., 2024). However, on-policy self-distillation (OPSD) avoids the need for a separate teacher model by using two conditioning branches of the same model: the student sees only the problem, while the teacher additionally receives privileged information such as a verified reference solution. On mathematical reasoning tasks, OPSD has been shown to be more token-efficient than reinforcement learning (Zhao et al., 2026a).

Making the rollout on-policy, however, does not make the teacher's supervision state-relative. The teacher remains conditioned on a static reference solution, which specifies the desired endpoint but does not explicitly represent what the student has established in its current rollout state. This matters when the rollout reaches a point with multiple valid continuations, while the reference represents only one of them. A reference-conditioned teacher may therefore assign lower probability to a valid alternative that departs from the reference trajectory, while offering limited guidance on whether a flawed intermediate state should be repaired or abandoned. Its supervision can consequently mix locally useful guidance with answer-dependent confidence and preference for a particular solution path (Li et al., 2026a; Kaur et al., 2026). As illustrated in Figure 1, a reference answer provides a fixed target trajectory, while the student's on-policy rollout reveals the reasoning state from which the next decision must be made. The limitation lies not in teacher capacity, but in conditioning: the teacher follows the reference path without explicit awareness of rollout-relative states, including viable alternatives and failure directions.

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.

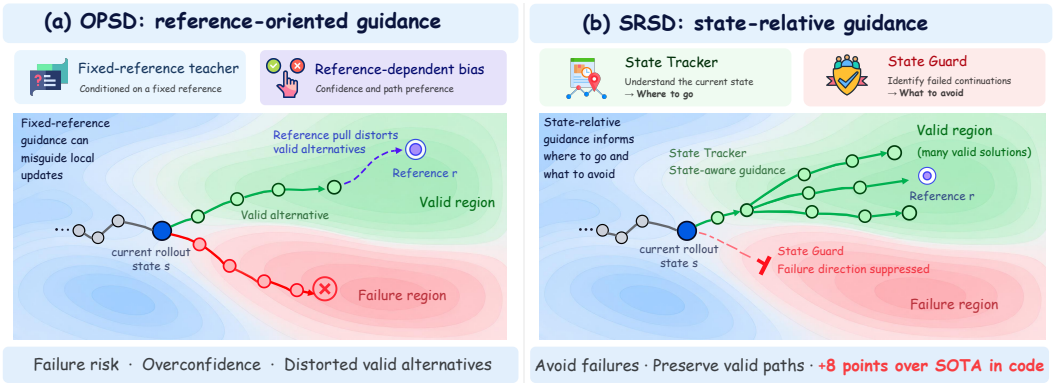

Figure 1: Reference-oriented guidance versus SRSD. (a) A teacher conditioned on the fixed reference r may penalize a valid alternative that departs from the reference path, while providing no explicit signal about the observed failure direction. (b) SRSD uses a State Tracker to extract viable guidance from rollouts and a State Guard to identify failure directions to avoid.

State-relative supervision should provide two complementary signals derived from the reasoning state induced by the current on-policy trajectory: *where to go* and *what to avoid*. We introduce **State-Relative Self-Distillation (SRSD)**, which augments reference-conditioned supervision with these two signals from the student's own on-policy rollouts. A positive teacher provides a viable direction for continuing or repairing the reasoning, while a negative teacher identifies a failure direction to avoid. Their contrast provides attractive and repulsive guidance at the same generated positions, reducing reliance on a single reference trajectory.

SRSD realizes this contrast through a **State Tracker** and a **State Guard**, detailed in Section 4. Here, state-relative refers to supervision conditioned on rollout-level state derived from complete on-policy trajectories, rather than a separately inferred state at each token position. The State Tracker identifies the reasoning state of the student's rollout to condition the positive teacher. Independently, the student samples a failed rollout from the same policy, which the State Guard uses as a failure reference for the negative teacher. SRSD then pulls the student toward viable paths while pushing it away from directions associated with observed failures. Our main contributions are as follows:

- We identify a limitation of existing OPSD-style methods: although student rollouts are on-policy, teacher supervision remains reference-conditioned, which can conflate locally useful guidance with answer-dependent confidence and preference for a particular solution path. We instead formulate supervision relative to the student's current rollout state.
- We propose SRSD, where privileged supervision is defined relative to the student's on-policy rollout rather than only to a fixed reference. SRSD realizes this formulation through a State Tracker and a State Guard, which construct complementary positive and negative teacher conditions within a unified attractive–repulsive objective.
- We evaluate SRSD on mathematical reasoning, code generation, and agentic tool use across multiple model scales and benchmarks. SRSD consistently outperforms all baselines, with the largest gain in code generation, where it exceeds the strongest baseline by 8 points. Extensive mechanistic analyses show that the two teacher signals become increasingly separable during training, while the student progressively shifts toward the positive teacher and away from the negative teacher.

2 Related Work

On-Policy Distillation. On-policy distillation (OPD) provides dense teacher supervision on trajectories sampled from the student's own policy. Given a student-generated prefix, the teacher scores each sampled token, and the teacher–student log-probability gap can serve as a token-level advantage for optimizing a reverse-KL objective through importance sampling (Lu & Lab, 2025; Agarwal et al., 2024). Existing OPD methods vary in feedback signal, teacher access, and loss granularity (Song & Zheng, 2026), but typically instantiate the teacher as a branch distinct from the student policy.

On-Policy Self-Distillation. On-policy self-distillation (OPSD) instead derives both student and teacher distributions from the same model, differentiating the teacher through privileged information such as reference solutions, verified trajectories, or additional context (Zhao et al., 2026a; Penaloza et al., 2026). Other variants construct supervision from environment feedback or outcome signals (Hübotter et al., 2026; He et al., 2026). Despite their different sources of supervision, these methods share a common structure: the student samples on-policy trajectories, while an information-augmented branch provides token-level guidance on the visited prefixes. Recent refinements improve privileged supervision along three main directions: *reliability*, *conditioning*, and *interpretation*. Reliability-oriented methods filter or regularize teacher signals (Yang et al., 2026; Xing et al., 2026), while conditioning-oriented methods enrich the teacher context with trajectory- or outcome-dependent information (Jiang & Ferraro, 2026; Xia et al., 2026b). RLCSD contrasts correct- and incorrect-hint conditioning to suppress shared style shifts and modulate verifier-derived GRPO advantages, rather than directly contrasting positive and negative teacher distributions for distillation (Pan et al., 2026). Interpretation-oriented studies examine how privileged context and solutions shape OPSD behavior (Li et al., 2026a; Ichihara et al., 2026).

Across these directions, existing work primarily improves how privileged supervision is constructed or understood, while remaining largely anchored to pre-specified privileged information. SRSD instead introduces rollout-relative state information and contrasts *where to go* with *what to avoid*, making supervision less dependent on reference-only guidance.

3 Problem Formulation

Given a problem x, the student policy π_θ generates a rollout $y = (y_1, \ldots, y_{|y|})$ autoregressively,

$$\pi_\theta(y \mid x) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid x, y_{<t}), \tag{1}$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the raw token prefix produced before step t. OPD performs distillation on complete rollouts $y \sim \pi_\theta(\cdot \mid x)$ sampled from the current student policy, while a separate teacher policy π_ϕ^T provides per-token distributions on the prefixes visited by the student. For a given rollout y, a generic on-policy distillation objective can therefore be written as

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{y\sim\pi_\theta(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D\big(\mathrm{sg}\big[\pi_\phi^T(\cdot \mid x, y_{<t})\big] \,\big\|\, \pi_\theta(\cdot \mid x, y_{<t})\big) \right]. \tag{2}$$

Here $\mathrm{sg}[\cdot]$ indicates that the teacher distribution is treated as a target with no gradient flowing through the teacher branch; D can be reverse KL, forward KL, JSD, or any computable distributional divergence. Unlike distillation from a fixed teacher trajectory, Eq. (2) supervises prefixes generated by the current student, aligning training with visited states (Agarwal et al., 2024). This requires the teacher to provide reliable feedback on student-generated prefixes.

OPSD preserves on-policy sampling but replaces the external teacher with a differently conditioned branch of the same model. Given a reference solution r, the student and teacher share the prefix $y_{<t}$, while the teacher additionally conditions on r. In our implementation, all branches share the same base model; only the student enables a single LoRA adapter, and we use π_θ to denote all branches for simplicity.

The positive teacher and the student of OPSD are two invocations of π_θ under different conditions,

$$q_t^+ := \pi_\theta(\cdot \mid x, r, y_{<t}), \qquad s_t := \pi_\theta(\cdot \mid x, y_{<t}), \tag{3}$$

and the corresponding distillation loss is

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \mathbb{E}_{(x,r)\sim\mathcal{D}}\, \mathbb{E}_{y\sim\pi_\theta(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D\big(\mathrm{sg}[q_t^+] \,\big\|\, s_t\big) \right], \tag{4}$$

where $D(\cdot\|\cdot)$ is a per-token divergence. Although the teacher observes the raw student prefix $y_{<t}$, the prefix does not explicitly identify which facts remain valid or whether an earlier error requires repair. Although standard OPSD can still correct an erroneous prefix, doing so requires the teacher to

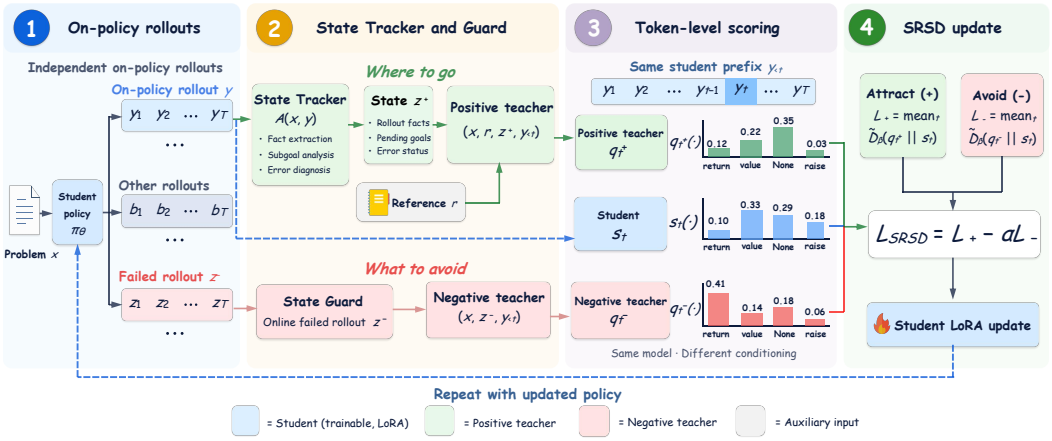

Figure 2: The SRSD training loop. The student independently samples on-policy trajectories from its current state. The State Tracker converts the supervised trajectory into a state representation for the positive teacher, while the State Guard provides a failed trajectory as a failure reference for the negative teacher. The two teachers score the same generated tokens, yielding attractive and repulsive signals that jointly update the student for the next round of on-policy sampling.

implicitly infer the student's reasoning state from the raw trajectory while remaining conditioned on the fixed reference, which brings the following problem. Such inference is itself challenging for process reward models and prompted critic models on process-level benchmarks (Zheng et al., 2025). Thus, OPSD is on-policy in the states visited by the student, while the privileged information used to construct the teacher distribution remains static. This motivates state-relative supervision that explicitly represents the student's current rollout state and contrasts viable and failed continuations, as introduced in Section 4.1.

4 State-Relative Self-Distillation

4.1 Method Overview

SRSD moves beyond purely reference-conditioned supervision by incorporating state-relative information from on-policy rollouts. We instantiate this principle with two complementary teacher conditions, as illustrated in Figure 2. Let $\mathcal{A}(x, y)$ denote an online analysis performed by the State Tracker on problem x and a complete student rollout y. The student first generates y according to Eq. (1); the State Tracker then reads x and y and returns the state representation

$$z^+ := \mathcal{A}(x, y). \tag{5}$$

State-relative supervision uses the student's on-policy rollout to infer its current reasoning state and conditions the teacher on this state. Here, state refers to the rollout-level reasoning condition induced by the complete trajectory, rather than an intermediate position within the rollout. The State Tracker is invoked once after the complete rollout y has been generated. It first determines an error-status marker $e \in \{\text{correct}, \text{wrong}, \text{trunc}\}$ and then applies the corresponding analysis prompt to produce the state representation z^+, which is shared across all scored token positions of y. Depending on e, the State Tracker identifies a continuation or repair direction. The resulting state representation z^+ encodes relevant rollout-level information, including established facts, unresolved subgoals, and unverified assumptions. We use DeepSeek-V4-Flash for this online state identification across all domains. When relying solely on the student model for rollout generation and diagnosis, performance remains competitive, with deviations from the default setting generally within 5%. Under the same settings with Table 1, only relying on student model also achieves SOTA-level performance on coding tasks. Details on the results and our motivation for using an external diagnosis model are provided in Appendix A.1.

Conditioned on z^+, the positive teacher extends the reference-conditioned teacher in Eq. (3) as

$$q_t^+(a) := \pi_\theta(a \mid x, r, z^+, y_{<t}), \tag{6}$$

where r is the verified reference solution. The additional state information allows the teacher to condition its guidance on an explicit summary of the reasoning state induced by the sampled trajectory. The student does not access z^+ and remains conditioned only on x and $y_{<t}$.

The State Guard provides the complementary negative direction through a failed rollout z^- sampled independently from the current policy. Unlike the State Tracker, it performs no additional analysis: z^- is passed directly to the negative teacher,

$$q_t^-(a) := \pi_\theta(a \mid x, z^-, y_{<t}). \tag{7}$$

Because z^- is sampled independently online from the same policy on the same problem, it reflects a current failure mode without requiring the same prefix or reasoning branch as the supervised rollout y. The negative teacher is not required to enumerate all possible failures; it suffices to expose failure modes expressed by the current policy. For tokens associated with such failures, when the negative teacher assigns higher probability than the student, the repulsive term suppresses the corresponding direction. Accordingly, the positive and negative teachers together yield complementary signals that specify *where to go* and *what to avoid*. The following subsection converts these signals into the SRSD training objective.

4.2 State-Relative Objective

SRSD compares three distributions at the same generated positions of y: the student s_t, the positive teacher q_t^+ of Eq. (6), and the negative teacher q_t^- of Eq. (7). We measure teacher–student discrepancy using forward KL divergence,

$$D(q_t \,\|\, s_t) := \mathrm{KL}(q_t \,\|\, s_t), \tag{8}$$

where q_t denotes either q_t^+ or q_t^-. We use the generalized JSD parameterization, where $\beta \in [0, 1]$ controls the interpolation from forward KL ($\beta = 0$) to reverse KL ($\beta = 1$); throughout SRSD, we set $\beta = 0$. For stable optimization, we apply element-level clipping to the vocabulary-wise KL contributions before aggregation, using a domain-specific threshold κ. We use $\kappa = 0$ to denote disabling element-level clipping; otherwise, each vocabulary-wise KL contribution is capped at κ.

Over valid generation positions $\mathcal{T}$, we define the positive attraction and negative reference losses as

$$\mathcal{L}_+ = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} D_\beta(q_t^+ \,\|\, s_t), \qquad \mathcal{L}_- = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} D_\beta(q_t^- \,\|\, s_t). \tag{9}$$

The SRSD objective combines attraction to the viable teacher with repulsion from the failure teacher,

$$\mathcal{L}_{\mathrm{SRSD}} = \mathcal{L}_+ - \alpha\, \mathcal{L}_-, \tag{10}$$

where α controls the strength of the failure-direction reference. Section 5.3 provides empirical evidence for this design. Training computes the losses in Eq. (9) over valid generation positions using the corresponding mask. Algorithm 1 summarizes one training step.

4.3 Analysis of State-Relative Contrast

We analyze the contrast between the two teacher signals directly in the probability space induced by the forward-KL objective, focusing on the unclipped objective. Let u_t denote the student logits at position t, such that $s_t = \mathrm{softmax}(u_t)$. For any teacher distribution q_t, the forward-KL gradient with respect to the student logits is

$$\nabla_{u_t} \mathrm{KL}(q_t \| s_t) = s_t - q_t. \tag{11}$$

Therefore, for the SRSD objective in Eq. (10),

$$-\nabla_{u_t} \mathcal{L}_{\mathrm{SRSD}} = \big(q_t^+ - s_t\big) - \alpha\big(q_t^- - s_t\big), \tag{12}$$

where the right-hand side is the direction followed by gradient descent.

For a candidate token a, define the positive and negative teacher residuals relative to the student as

$$\Delta_t^+(a) := q_t^+(a) - s_t(a) = g_t(a) + h_t(a), \tag{13}$$

and

$$\Delta_t^-(a) := q_t^-(a) - s_t(a) = b_t(a) + h_t(a). \tag{14}$$

Algorithm 1 State-Relative Self-Distillation (one training step)

Require: problem x, reference r, policy π_θ, rollouts $K \geq 2$, clip threshold κ, repulsion weight α
Ensure: updated parameters θ

1: $\{y^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid x)$; select rollout y and a distinct failed rollout z^- ▷ on-policy rollouts
2: $z^+ \leftarrow \mathcal{A}(x, y)$ ▷ State Tracker
3: $\mathcal{T} \leftarrow$ valid positions of y
4: $s_t \leftarrow \pi_\theta(\cdot \mid x, y_{<t}), \quad t \in \mathcal{T}$ ▷ student branch
5: $q_t^+ \leftarrow \mathrm{sg}[\pi_\theta(\cdot \mid x, r, z^+, y_{<t})]$ ▷ positive teacher
6: $q_t^- \leftarrow \mathrm{sg}[\pi_\theta(\cdot \mid x, z^-, y_{<t})]$ ▷ negative teacher
7: $\delta_t^\sigma(a) \leftarrow q_t^\sigma(a)\big(\log q_t^\sigma(a) - \log s_t(a)\big), \quad \sigma \in \{+, -\}$
8: $\widetilde{D}_{\mathrm{KL}}(q_t^\sigma \| s_t) \leftarrow \sum_a \begin{cases} \delta_t^\sigma(a), & \kappa = 0, \\ \min\{\delta_t^\sigma(a), \kappa\}, & \kappa > 0, \end{cases} \quad \sigma \in \{+, -\}$ ▷ element-level clipping
9: $\mathcal{L}_\sigma \leftarrow \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \widetilde{D}_{\mathrm{KL}}(q_t^\sigma \| s_t), \quad \sigma \in \{+, -\}$
10: update θ by minimizing $\mathcal{L}_+ - \alpha \mathcal{L}_-$ ▷ SRSD objective

Here, $g_t(a)$ denotes the component associated with the rollout-level state information provided to the positive teacher, $b_t(a)$ denotes the component associated with the failed rollout z^-, and $h_t(a)$ denotes a component shared by the two teacher residuals. Such a shared component can capture probability shifts induced by conditioning effects that are common to both teacher branches rather than specific to either directional signal.

Substituting Eq. (13) and Eq. (14) into the exact gradient-descent direction in Eq. (12) gives

$$\Delta_{\mathrm{SRSD},t}(a) = \Delta_t^+(a) - \alpha\, \Delta_t^-(a) = g_t(a) - \alpha\, b_t(a) + (1 - \alpha)\, h_t(a). \tag{15}$$

Thus, when $\alpha = 1$, any component shared by the two probability residuals cancels exactly,

$$\Delta_{\mathrm{SRSD},t}(a) = g_t(a) - b_t(a). \tag{16}$$

For $0 < \alpha < 1$, the shared component remains but is scaled by $(1-\alpha)$. This cancellation is therefore a property of the forward-KL gradient itself, rather than of a separate log-ratio approximation.

Eq. (15) characterizes the interaction between the two teachers. Residuals with the same sign partially cancel under the repulsive term, whereas residuals with opposite signs reinforce the same update direction. SRSD therefore places greater emphasis on tokens where the two conditioning branches disagree directionally.

If the two teacher residuals contain a shared component h_t, contrasting them attenuates the effects common to both branches while preserving the difference that depends on the state. This enables SRSD to steer away from observed failures without incurring a penalty for departing from the reference solution.

5 Experiments

Our experiments answer three research questions: (1) Does SRSD consistently outperform OPSD and related baselines across diverse settings (Section 5.2)? (2) How do the contrast teachers contribute to SRSD, and how does the repulsion weight affect their interaction (Section 5.3)? (3) How does the state-relative contrast evolve during training, and does the student progressively shift from the negative toward the positive teacher (Section 5.4)?

5.1 Experimental Setup

Mathematical reasoning. We train Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Yang et al., 2025) on a 30k-problem mathematical subset of OpenThoughts (Guha et al., 2025). We evaluate on AIME24, AIME25, and HMMT25, reporting Avg@12 for each benchmark. We compare against the base model, supervised fine-tuning, GRPO (Shao et al., 2024), and OPSD (Zhao et al., 2026a), together with recent on-policy self-distillation methods evaluated at the same three model scales: AD-OPSD (Peng et al., 2026), PHF (Li et al., 2026c), PS-OPSD (Zhao et al., 2026b), ATESD (Han et al., 2026), SGSD (Huang et al., 2026), DAPD (Wu et al., 2026), AVSD (Nguyen et al., 2026), RLCSD (Pan et al., 2026), and u-OPSD (Li et al., 2026b). Baseline results are taken from the corresponding source papers.

Table 1: Main results on coding and agentic tool use. We report pass@1 on LiveCodeBench v5/v6 and accuracy on the four BFCL v3 subsets: Base, Missing Function (MF), Missing Parameter (MP), and Long Context (LC). **Avg** denotes the mean across the columns. Baseline results are from (Zhang et al., 2026). Our method is shaded; best and second-best results are **bolded** and underlined.

	Coding (LiveCodeBench)			Agentic tool use (BFCL v3)				
Method	**Avg**	v5	v6	**Avg**	Base	MF	MP	LC
		Qwen3-4B				*Qwen3-4B*		
Base	44.45	47.67	41.22	22.88	28.50	20.50	20.50	22.00
GRPO	47.23	50.18	44.27	25.25	**33.00**	20.50	**24.50**	23.00
Skill-SD	45.90	49.82	41.98	24.63	**33.00**	19.00	24.00	22.50
OPSD	38.88	42.65	35.11	25.13	31.50	19.50	20.50	**29.00**
SDPO	37.61	40.86	34.35	15.75	21.50	14.00	13.50	14.00
SDFT	46.13	48.75	43.51	21.75	31.50	17.00	17.00	21.50
LOPD	47.79	50.54	45.04	27.38	32.50	26.50	**24.50**	26.00
SRSD	**55.72**	**58.78**	**52.67**	**28.25**	**33.00**	**29.00**	23.50	27.50
		Olmo3-7B				*Qwen3-8B*		
Base	45.79	47.31	44.27	28.38	34.00	28.50	25.50	25.50
GRPO	47.63	49.46	45.80	29.00	36.00	28.00	27.00	25.00
Skill-SD	47.07	49.10	45.03	27.38	33.00	27.00	25.00	24.50
OPSD	44.57	44.09	45.04	25.75	30.50	25.00	23.50	24.00
SDPO	46.67	49.82	43.51	25.00	29.00	24.50	23.00	23.50
SDFT	45.97	47.67	44.27	26.88	31.50	26.00	24.50	25.50
LOPD	49.61	53.41	45.80	29.88	37.00	29.00	**27.50**	**26.00**
SRSD	**55.35**	**57.25**	**53.44**	**30.38**	**38.50**	**31.00**	27.00	25.00

Code generation. We evaluate code generation on LiveCodeBench (Jain et al., 2025) v5 and v6, reporting pass@1 separately for each split. Training uses the verified TACO (Li et al., 2023) split released with DeepCoder (Luo et al., 2025). We conduct experiments on two model families, Qwen3-4B (Yang et al., 2025) and Olmo3-7B (Olmo et al., 2025). We compare against the base model, GRPO (Shao et al., 2024), Skill-SD (Wang et al., 2026), OPSD (Zhao et al., 2026a), SDPO (Hübotter et al., 2026), SDFT (Shenfeld et al., 2026), and LOPD (Zhang et al., 2026), using the results reported by (Zhang et al., 2026).

Agentic tool-use. We evaluate multi-turn tool use on BFCL v3 (Patil et al., 2025) with Qwen3-4B and Qwen3-8B (Yang et al., 2025). Training uses the EnvScaler (Song et al., 2026) training split, which contains programmatically synthesized tool environments with rule-based validators for complete trajectories. We report accuracy on the four BFCL v3 multi-turn subsets: Base, Missing Function (MF), Missing Parameter (MP), and Long Context (LC). We use the same comparison methods and reported results as in the coding experiments (Zhang et al., 2026).

The student and both teachers share the same base model, with a single LoRA adapter enabled only for the student. Unless stated otherwise, we set $\beta = 0$, reducing the positive and negative terms to $\mathrm{KL}(q_t^+ \| s_t)$ and $\mathrm{KL}(q_t^- \| s_t)$. We compute the loss over the full vocabulary rather than restricting it to either teacher's top-k support. Across all runs, we use a learning rate of 5×10^{-6}, an effective batch size of 32, a maximum gradient norm of 0.1, and 200 training steps. Appendix B provides the dataset filtering and remaining domain-specific training and evaluation details, while Appendix D lists the prompts used during training.

5.2 Main Results

Table 1 reports results on code generation and agentic tool use across multiple backbones. SRSD shows its largest gains in code generation. On Qwen3-4B, it achieves 58.78 and 52.67 pass@1 on LiveCodeBench v5 and v6, respectively, for an average of 55.72. This improves over the strongest baseline, LOPD (47.79), by 7.9 points and over Base (44.45) by 11.27 points. The gain remains substantial on Olmo3-7B, where SRSD reaches 55.35 on average, exceeding LOPD (49.61) by 5.74 points. In the agentic tool-use domain, SRSD achieves the highest average performance on BFCL

Table 2: Main results on mathematical reasoning across the Qwen3 model family. We report Avg@12 on AIME24 (A24), AIME25 (A25), and HMMT25 (H25), with **Avg** denoting their mean. "–" denotes unavailable results. Our method is shaded. Best and second-best results are **bolded** and underlined, respectively.

	Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
Method	**Avg**	A24	A25	H25	**Avg**	A24	A25	H25	**Avg**	A24	A25	H25
Base	37.10	51.50	36.70	23.10	61.20	74.90	66.40	42.20	61.80	75.80	65.60	43.90
SFT	35.80	48.40	36.30	22.70	58.60	70.20	62.30	43.40	59.80	72.30	64.20	42.90
GRPO	37.70	51.10	38.30	23.70	62.70	75.60	68.10	44.40	64.00	76.40	68.90	46.70
OPSD	43.40	57.20	43.90	29.20	63.60	76.40	68.30	46.10	64.80	77.80	70.80	45.80
AD-OPSD	38.70	50.60	40.60	25.00	61.80	74.40	66.40	44.70	–	–	–	–
PHF	43.06	58.89	41.67	28.61	63.39	76.11	68.06	46.00	65.94	77.78	71.72	48.33
PS-OPSD	43.12	55.37	41.48	**32.50**	64.32	76.57	70.56	45.83	65.40	77.13	71.30	47.78
ATESD	44.35	59.17	44.72	29.17	65.65	**78.06**	71.39	47.50	67.13	**80.56**	72.50	48.33
SGSD	43.70	57.80	43.60	29.70	64.30	75.30	70.80	46.70	65.50	78.90	70.60	46.90
DAPD	43.98	57.50	**46.67**	27.78	65.28	77.22	**72.22**	46.39	67.41	80.00	72.50	49.72
AVSD	39.01	51.85	38.61	26.57	62.78	75.46	66.67	46.20	63.64	77.78	66.67	46.48
RLCSD	–	53.10	38.30	–	–	74.40	69.40	–	–	77.50	69.70	–
u-OPSD	–	–	–	–	63.80	76.39	68.06	46.94	65.28	76.94	71.39	47.50
SRSD	**45.65**	**59.44**	45.56	31.94	**66.00**	77.22	71.94	**48.83**	**68.61**	79.72	**75.83**	**50.28**

Table 3: Ablation of the State Tracker, State Guard, and repulsion weight α. We report the mean Avg@12 across AIME24, AIME25, and HMMT25. Tracker only uses $\alpha = 0$ and retains $\mathcal{L}_+$ alone, while Guard only removes $\mathcal{L}_+$ and retains $-\alpha\mathcal{L}_-$ with $\alpha = 0.3$. Best result in each row is **bolded**.

		State ablation		α ablation		
Model	**Base**	**Tracker only** ($\alpha = 0$)	**Guard only**	$\boldsymbol{\alpha = 0.3}$	$\alpha = 0.6$	$\alpha = 1.0$
Qwen3-1.7B	37.10	42.98	35.19	**45.65**	44.54	43.24
Qwen3-4B	61.20	63.89	58.23	**66.00**	65.00	65.37
Qwen3-8B	61.80	67.22	60.16	**68.61**	68.15	66.11

v3 for both Qwen3-4B and Qwen3-8B, reaching 28.25 and 30.38, respectively. Its clearest gains occur on Missing Function, where it improves over LOPD from 26.50 to 29.00 on Qwen3-4B and from 29.00 to 31.00 on Qwen3-8B.

Table 2 compares SRSD with OPSD and other on-policy self-distillation methods across three Qwen3 scales. SRSD achieves the best average accuracy over AIME24, AIME25, and HMMT25 at all three scales, reaching 45.65, 66.00, and 68.61 on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively. The improvement is most pronounced on Qwen3-8B, where SRSD raises the average from 67.41 for the strongest baseline to 68.61, while achieving 75.83 on AIME25 compared with 72.50 for the strongest baseline and 65.60 for the base model. Overall, SRSD achieves the highest average across all evaluated backbone–domain combinations, with particularly pronounced improvements in code generation.

5.3 Ablation of the Two Teachers and the Repulsion Strength α

Table 3 ablates SRSD along two axes: removing either teacher and varying the repulsion weight α with both teachers active. The single-teacher results show that the attractive and repulsive terms in Eq. (10) play distinct roles.

Tracker only sets $\alpha = 0$ and retains only $\mathcal{L}_+$. It recovers most of the improvement over Base, with gains of $+5.88$, $+2.69$, and $+5.42$ points at 1.7B, 4B, and 8B, respectively. In contrast, Guard only retains $-\alpha\mathcal{L}_-$ and consistently falls below Base by 1.91, 2.97, and 1.64 points. This asymmetry follows from the objective: minimizing $-\alpha\mathcal{L}_-$ only pushes the student away from the negative teacher, without specifying which alternative direction is desirable. These results support using the repulsive term as a directional correction to positive guidance rather than as a standalone objective.

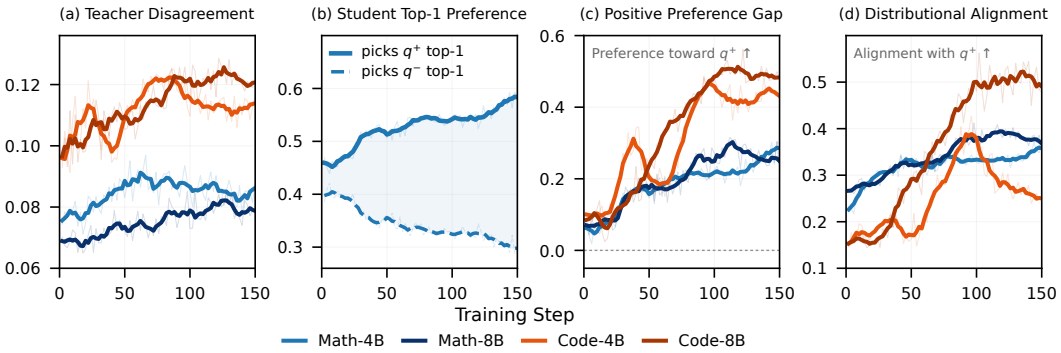

Figure 3: Training dynamics of SRSD. (a) **Teacher Disagreement**: fraction of positions where the positive and negative teachers prefer different top-1 tokens ($\arg\max q^+ \neq \arg\max q^-$). (b) **Student Top-1 Preference**: among disagreement positions, the fraction where the student's top-1 token matches the positive (solid) or negative (dashed) teacher. (c) **Positive Preference Gap**: difference between the two agreement rates in (b). (d) **Distributional Alignment**: cosine similarity between the teacher contrast and the student's displacement from the negative teacher, $\cos(q^+ - q^-,\, s - q^-)$. Thin lines show per-window values; thick lines show five-window moving averages.

With both teachers active, $\alpha = 0.3$ performs best at every scale, reaching 45.65, 66.00, and 68.61 at 1.7B, 4B, and 8B. Increasing α further consistently degrades performance. Although $\alpha = 1.0$ exactly cancels the assumed shared component h_t in Eq. (15), it also gives the repulsive term equal weight to the attractive signal, reducing accuracy by 2.41, 0.63, and 2.50 points relative to $\alpha = 0.3$. These results favor partial rather than exact cancellation, and we therefore use $\alpha = 0.3$ for mathematical reasoning. Coding and agentic tool-use instead use $\alpha = 0.6$, with the full domain-specific configuration reported in Appendix B.4.

5.4 Mechanistic Analysis

SRSD relies on a meaningful contrast between the two teachers. If they produced identical distributions, Eq. (10) would reduce to $(1-\alpha)\mathcal{L}_+$, leaving only a rescaled positive loss. We therefore examine whether the teachers become increasingly distinguishable during training and whether the student preferentially follows the positive teacher. Figure 3 tracks across four configurations, covering Qwen3-4B and Qwen3-8B on math and coding tasks.

Figure 3(a) shows that teacher disagreement rises over training, indicating that the positive and negative conditioning branches become more separable. At these disagreement positions, the student increasingly favors the positive teacher's top-1 token while reducing its preference for the negative teacher's choice (Figure 3(b)). This asymmetric shift yields a substantially larger positive preference gap across all four runs, consistently across model scales and domains (Figure 3(c)). Figure 3(d) further shows that this trend extends beyond top-1 predictions: the cosine similarity remains positive and generally increases over training, indicating that the student's overall distribution increasingly shifts from the negative toward the positive teacher. Together, these results show a growing teacher contrast accompanied by a consistent student shift toward the positive direction. Additional mechanistic analyses and further discussion are provided in Appendix A.2.

6 Conclusion

Existing on-policy self-distillation can still rely on static reference information for teacher supervision. We introduced SRSD, where a State Tracker provides the positive *where to go* signal from the student's rollout state, while a State Guard provides the negative *what to avoid* signal from observed failures. Across mathematical reasoning, code generation, and agentic tool use, SRSD consistently improves over OPSD-style baselines across model scales, with gains of up to 8 points in code generation. Mechanistic analyses further show that the two teacher signals become increasingly separable during training, while the student progressively shifts toward the positive teacher and away from the negative teacher. Together, these results support state-relative teacher supervision as an effective alternative to relying primarily on static reference guidance.

AI USE STATEMENT

We used generative AI tools to lightly polish the wording of the manuscript, to assist in locating related work during the literature search, and for figure conceptualization. All AI-assisted text and retrieved references were reviewed by the authors, and we take responsibility for the final content of this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. OpenThoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Zihao Han, Tiangang Zhang, Huaibin Wang, and Yilun Sun. Adaptive teacher exposure for self-distillation in LLM reasoning. *arXiv preprint arXiv:2605.11458*, 2026.

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

Jiazhen Huang, Xiao Chen, Xiao Luo, Yong Dai, Senkang Hu, and Yuzhi Zhao. Skill-conditioned gated self-distillation for LLM reasoning. *arXiv preprint arXiv:2605.28791*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Yuki Ichihara, Naoto Iwase, Mohammad Atif Quamar, and Junpei Komiyama. Privileged solutions or context-induced teacher behavior? dissecting on-policy self-distillation. *arXiv preprint arXiv:2608.09228*, 2026.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Yuxuan Jiang and Francis Ferraro. Bridging reasoning trajectories in on-policy distillation via near-future guidance. *arXiv preprint arXiv:2606.00305*, 2026.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026a.

Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision. *arXiv preprint arXiv:2608.06296*, 2026b.

Yuhan Li, Mingxu Zhang, Dazhong Shen, and Ying Sun. PHF: Privileged hidden flow for on-policy self-distillation. *arXiv preprint arXiv:2606.29340*, 2026c.

Hao Lin, Kunyang Lv, Xu Jiang, Jingqi Tian, Zhongjing Du, Jiayu Ding, Qiaoman Zhang, and Hongbo Jin. Visd: Enhancing video reasoning via structured self-distillation. *arXiv preprint arXiv:2605.06094*, 2026.

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

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349 2025. Notion Blog.

Duy Nguyen, Hanqi Xiao, Archiki Prasad, Zaid Khan, Anirban Das, Austin Zhang, Sambit Sahu, Hyunji Lee, Elias Stengel-Eskin, and Mohit Bansal. AVSD: Adaptive-view self-distillation by balancing consensus and teacher-specific privileged signals. *arXiv preprint arXiv:2605.20643*, 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. RLCSD: Reinforcement learning with contrastive on-policy self-distillation. *arXiv preprint arXiv:2606.11709*, 2026.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Keqin Peng, Chen Li, Yuanxin Ouyang, Yancheng Yuan, and Liang Ding. Diagnosing and mitigating thinking collapse in on-policy self-distillation. *arXiv preprint arXiv:2607.10805*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis. *arXiv preprint arXiv:2601.05808*, 2026.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-sd: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026.

Jianyu Wu, Yizhou Wang, Encheng Su, Chen Tang, and Shixiang Tang. DAPD: Dual-anchored policy distillation. *arXiv preprint arXiv:2608.01735*, 2026.

Mingxuan Xia, Yuhang Yang, Chao Ye, Shuai Zhu, Shenzhi Yang, Guangcheng Zhu, Yuhang Zhang, Cheng Peng, Haobo Wang, and Siqing Wang. Enhancing rubric-based rl via self-distillation. *arXiv preprint arXiv:2607.18082*, 2026a.

Xu Xia, Jinghua Piao, Min Yang, Xiaochong Lan, Jiaju Chen, and Yong Li. From scoring to acting: Outcome-verified comparative self-distillation for LLM agents. *arXiv preprint arXiv:2607.27937*, 2026b.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

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

Zhicheng Yang, Zhijiang Guo, Yifan Song, Minrui Xu, Yongxin Wang, Yiwei Wang, Xiaodan Liang, and Jing Tang. Prune-OPD: Efficient and reliable on-policy distillation for long-horizon reasoning. *arXiv preprint arXiv:2605.07804*, 2026.

Guibin Zhang, Jiayang Lyu, Ran Sun, Xinlei Yu, Haoyu Zhao, Qibing Ren, and Shuicheng Yan. Latent on-policy self-distillation. *arXiv preprint arXiv:2608.13040*, 2026.

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*, 2026a.

Xuyang Zhao, Liting Zhang, Zichen Xu, Zhihu Wang, Caiyue Xu, Shiwan Zhao, and Qicheng Li. Is more privileged information better? from solution traces to problem-solving structure in self-distilled reasoning. *arXiv preprint arXiv:2608.01589*, 2026b.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

APPENDIX OVERVIEW

This appendix is organized into five parts. Appendix A reports additional ablations and supplementary results beyond the main paper. Appendix B details the data, training, and evaluation configuration used in each of the three application domains. Appendix C discusses the main limitations of SRSD. Appendix D lists the exact prompt templates issued to the student and the two teachers within a single training step. Appendix E presents two worked case studies comparing OPSD and SRSD generations on AIME24 problems.

- **Appendix A: Additional Ablation and Results.**
  - *A.1 Effect of the Diagnosis Model in Online Rollout Diagnosis.* Ablates which model produces the State Tracker's diagnosis (Self-Model, Self-Model + Self-Correct, or DeepSeek-V4-Flash) against a stronger static-reference control on code generation, showing the gain comes from rollout-relative diagnosis rather than reference strength alone.
  - *A.2 Supplementary Evidence for Mechanistic Analysis.* Extends the training-dynamics analysis of Section 5.4 across more models, domains, and repulsion strengths, and adds a token-localized study showing teacher divergence concentrates at self-correction ("pivot") tokens.
  - *A.3 Ablation on the Positive Teacher's Conditioning Content.* Ablates what is placed in the positive teacher's prompt (reference solution vs. diagnosis vs. self-corrected rollout), isolating how much of SRSD's gain comes from the dual-teacher structure itself versus the specific conditioning content.
  - *A.4 Hyperparameter Ablation on Mathematical Reasoning.* Sweeps the JSD mixture coefficient β, learning rate, rollout temperature, and max completion length on Qwen3-1.7B mathematical reasoning.
  - *A.5 α/κ Ablation on Code and Agentic Tool Use.* Extends the repulsion weight α / clipping threshold κ sweep from the main paper to coding and agentic tool use.
- **Appendix B: Data, Training, and Evaluation.** Describes the shared State Tracker error-status scheme and rollout filtering, then gives the per-domain training and evaluation configuration for mathematical reasoning, coding, and agentic tool use.
- **Appendix C: Limitations.** Discusses the limitations of SRSD, including the additional inference cost introduced by online state analysis, the dependence on state extraction quality, the coverage of sampled failure modes, and the scalability to larger models and broader task settings.
- **Appendix D: Prompt Templates.** Shows the exact prompts issued to the student, the State Tracker, and the positive/negative teachers, and how the shared generated trajectory is scored across the three branches.
- **Appendix E: Case Studies.** Compares OPSD and SRSD generations on two AIME24 problems, one where a localized algebraic error separates the two methods and one where both reach the correct answer via distinct valid derivations.

A ADDITIONAL ABLATION AND RESULTS

A.1 EFFECT OF THE DIAGNOSIS MODEL IN ONLINE ROLLOUT DIAGNOSIS

The State Tracker's online diagnosis of the supervised rollout (Section 4.2) can be produced by different diagnosis models. We ablate this choice on code generation using Qwen3-4B and Olmo-7B. Besides the Base, OPSD, and LOPD baselines, we consider a stronger static-reference control, *OPSD + Static Answer*, where DeepSeek-V4-Flash generates a stronger fixed answer without rollout-relative diagnosis. For SRSD, we compare three diagnosis settings. *Self-Model* uses the student model itself as the State Tracker to produce the state diagnosis from the supervised rollout. *Self-Model + Self-Correct* additionally provides a self-corrected rollout as auxiliary context to the State Tracker when producing the diagnosis. The self-corrected rollout is used only to assist diagnosis and is not directly provided to the positive teacher. *DeepSeek-V4-Flash* uses DeepSeek-V4-Flash as the State Tracker and corresponds to the default setting used in the main experiments. Table 4 reports the average pass@1 over LiveCodeBench v5 and v6.

Table 4: Effect of the diagnosis model used for state-relative supervision on code generation. Each entry reports the average pass@1 over LiveCodeBench v5 and v6. Self-Model uses the student model as the State Tracker to produce the state diagnosis from the supervised rollout. Self-Model + Self-Correct additionally provides a self-corrected rollout only as auxiliary context to the State Tracker when producing the diagnosis; it is not directly used to condition the positive teacher. DeepSeek-V4-Flash serves as the State Tracker in the default SRSD setting. For comparison, OPSD + Static Answer replaces the standard reference with a stronger static answer generated by DeepSeek-V4-Flash, without introducing rollout-relative state diagnosis. Best and second-best results within each model are **bolded** and underlined, respectively. The shaded column denotes the default SRSD setting.

Model	Baselines			Static Reference	SRSD		
	Base	OPSD	LOPD	OPSD + Static Answer DeepSeek-V4-Flash	Self-Model diagnosis	Self-Model diagnosis + self-correct context	DeepSeek-V4-Flash default SRSD
Qwen3-4B	44.45	38.88	47.79	39.61	48.73	<u>53.42</u>	**55.72**
Olmo-7B	45.79	44.57	49.61	45.38	50.87	<u>53.59</u>	**55.35**

Static reference versus rollout-relative diagnosis. Table 4 separates the effect of stronger static supervision from that of rollout-relative diagnosis. Replacing the OPSD reference with a static answer generated by DeepSeek-V4-Flash yields only small improvements over OPSD, from 38.88 to 39.61 on Qwen3-4B and from 44.57 to 45.38 on Olmo-7B, while remaining below the corresponding Base models. In contrast, all SRSD variants outperform both OPSD and the stronger static-reference control. Even Self-Model, which uses no external diagnosis model, reaches 48.73 and 50.87 and exceeds LOPD on both backbones. These results indicate that the gain does not arise simply from replacing the reference with a stronger static answer or from access to a stronger external model.

Self-corrected context for the State Tracker. Providing a self-corrected rollout as additional context to the self-model State Tracker further improves the average LiveCodeBench score to 53.42 on Qwen3-4B and 53.59 on Olmo-7B. The self-corrected rollout is used only to assist the State Tracker in producing the diagnosis and is not directly used as the conditioning trajectory of the positive teacher. Compared with the default DeepSeek-V4-Flash setting, Self-Model + Self-Correct is only 2.30 and 1.76 points lower on Qwen3-4B and Olmo-7B, corresponding to relative deviations of 4.1% and 3.2%, respectively. These results show that SRSD does not critically depend on an external analyzer. We use DeepSeek-V4-Flash in the main experiments for its favorable performance–cost trade-off, providing stronger diagnostic context with modest additional training-time overhead.

Justification for an external diagnosis model. Introducing an external model into the distillation loop is not something OPSD-style methods need to avoid. Several recent on-policy self-distillation variants deliberately bring in an independent model precisely because it improves performance: VISD uses an external LLM judge to generate structured correctness, consistency, and grounding feedback that conditions the self-teacher (Lin et al., 2026); Skill-SD uses an auxiliary LLM to asynchronously summarize completed episodes into skill text injected into the self-teacher prompt (Wang et al., 2026); and CriPO uses an external rubric judge to score rollouts and localize the criterion-injection self-teacher (Xia et al., 2026a). In all three cases the external model only supplies text-level feedback, while the token-level distillation targets still come from a self-teacher sharing the student's own weights, consistent with our own default setting. We adopt DeepSeek-V4-Flash as the State Tracker for the same reason – it improves performance over a purely self-model diagnosis, as shown above. For a fair comparison against methods that introduce no external model at all, we also report the Self-Model and Self-Model + Self-Correct settings, which use no external diagnosis model yet still outperform both OPSD and LOPD.

A.2 Supplementary Evidence for Mechanistic Analysis

A.2.1 Consistency of Teacher Contrast and Student Directional Shift

The analysis in Section 5.4 tracks four runs under the corresponding setting. To examine whether the observed dynamics extend beyond this specific configuration, we conduct an additional diagnostic study spanning Qwen3-4B and Qwen3-8B, mathematical reasoning and coding, multiple repulsion

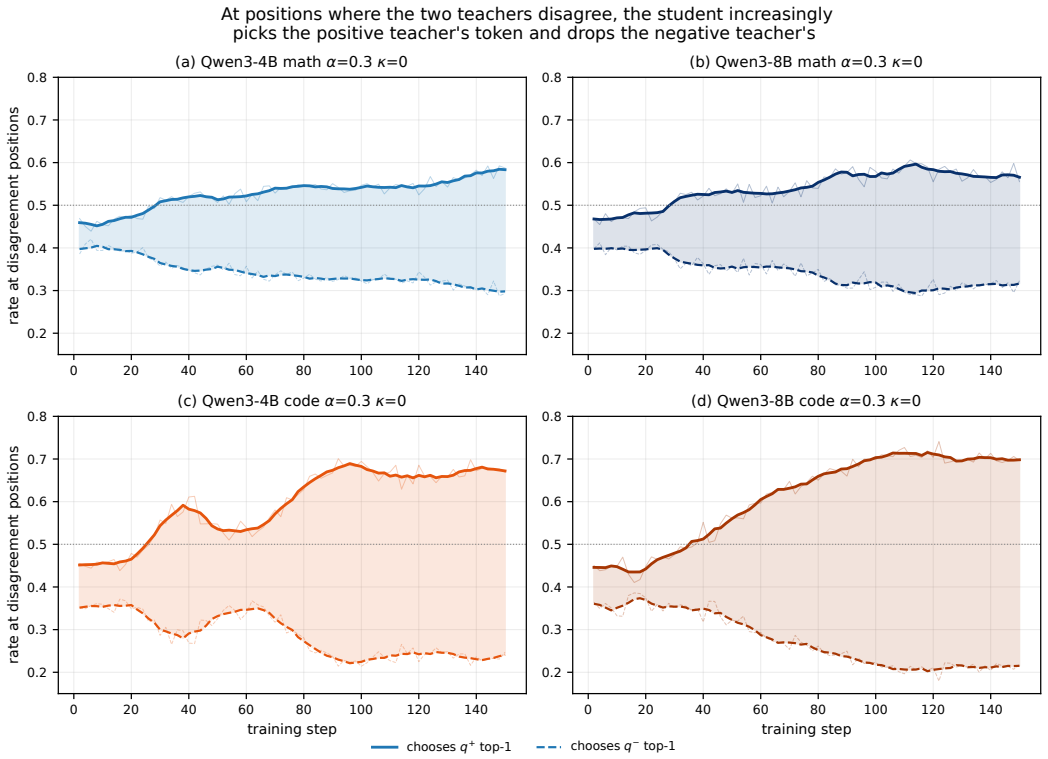

Figure 4: Per-run evolution of the positive preference gap. Each panel corresponds to one of the four runs with $\alpha = 0.3$, $\kappa = 0$, which are the same runs used in Figure 3: (a) Qwen3-4B mathematical reasoning; (b) Qwen3-8B mathematical reasoning; (c) Qwen3-4B coding; (d) Qwen3-8B coding. Solid and dashed lines denote the fractions of teacher-disagreement positions where the student's top-1 token matches the positive and negative teacher, respectively. The shaded region represents their difference. Thin curves show per-window measurements and thick curves show five-window moving averages. Across all four runs, the preference gap increases consistently over training, from 0.079–0.127 at the beginning to 0.257–0.490 at the end. The reported Cliff's $\delta = +1.00$ is a non-parametric effect size measuring the direction and consistency of this increase across training windows, indicating that later windows consistently exhibit larger gaps than earlier ones.

strengths, and element-level clipping thresholds. Similar to Section 5.4, this study investigates two questions: (i) whether the positive and negative teacher distributions become increasingly distinguishable during training, and (ii) whether the student consistently shifts toward the positive teacher and away from the negative teacher across models, domains, and training configurations.

Teacher contrast versus confidence. We first verify that the teacher contrast reflects different supervision signals rather than merely different confidence levels. Across all candidates, the confidence difference between the two teachers remains small, while their token-level predictions exhibit substantially larger divergence. Specifically, the entropy difference $H(q^+) - H(q^-)$ stays below 0.073 across all runs, whereas the divergence between the two teacher distributions is consistently larger, with $\mathrm{KL}(q^- \| q^+)$ exceeding the entropy gap by a median factor of 3.5–19.9 per training window. In addition, the symmetric divergence between the two teachers remains positive throughout all 900 logged windows, with candidate-level averages ranging from 0.015 to 0.047. These results show that the two teachers maintain distinct token preferences during training rather than collapsing into equivalent supervision signals.

Token-level directional shift. Figure 4 further examines this directional shift at individual token decisions. The analysis only considers positions where the two teachers already disagree on their top-1 prediction. Therefore, the widening gap cannot be explained by a global increase in student

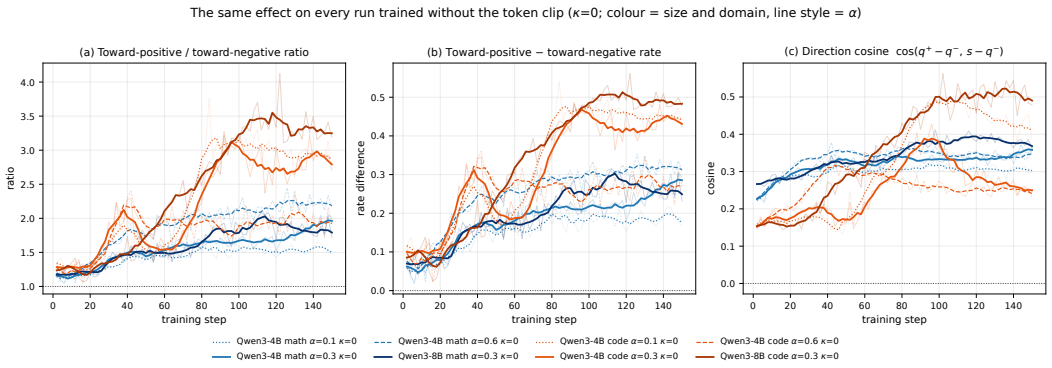

Figure 5: Robustness of student directional preference across repulsion strengths. The figure aggregates all eight candidates with $\kappa = 0$, covering $\alpha \in \{0.1, 0.3, 0.6\}$, two model sizes, and two domains. (a) **Preference ratio**: the ratio between the fractions of disagreement positions matching the positive and negative teachers. (b) **Preference margin**: the difference between these two fractions, following Figure 3(c). (c) **Distributional alignment**: cosine similarity between the teacher contrast direction and the student's displacement from the negative teacher, following Figure 3(d). Across the eight candidates, all three metrics exhibit consistent directional changes over training.

confidence, which would influence both teachers similarly. Instead, it reflects a relative shift in the student's preference toward the positive teacher and away from the negative teacher. This pattern appears consistently across all four model-size–domain combinations.

Source of the increasing teacher contrast. Importantly, the increasing teacher contrast does not result from updating the teachers: both branches remain frozen and share the same base model. Instead, training changes the student policy and therefore the on-policy rollouts from which z^+ and z^- are constructed. As the student improves, the positive branch is increasingly conditioned on rollouts with clearer established facts, active subgoals, and viable next steps, while the failed rollouts used by the negative branch more directly expose the residual failure modes of the current policy. The two frozen teachers are therefore evaluated under increasingly distinct positive and negative conditioning contexts, which can lead to greater separation between their output distributions even though the teacher weights themselves remain fixed.

Robustness across repulsion strengths. Figure 5 extends the analysis across the swept range of α with $\kappa = 0$. The three metrics characterize the same training dynamics from complementary perspectives: an argmax-based ratio, an absolute preference difference, and a distribution-level directional alignment. Despite capturing different aspects of the student update, all three exhibit consistent trends across the eight runs and 600 logged windows. These observations indicate that the directional preference identified in Section 5.4 is not specific to the default $\alpha = 0.3$ setting, but remains observable across different repulsion strengths $\alpha \in \{0.1, 0.3, 0.6\}$.

Overall, this supplementary study provides additional evidence that the mechanism of SRSD is stable beyond the default configuration. Across models, domains, and training settings, the positive and negative teachers remain distinguishable, and the student consistently resolves their disagreement toward the positive direction. These findings support the interpretation in Section 5.4 that SRSD learns from the contrast between two state-dependent signals rather than relying on a single teacher preference.

A.2.2 Token-Localized Evidence at Pivot Points

The training-level analyses above aggregate divergence uniformly across all generated positions. We complement them with a token-localized, inference-time study that asks whether the teacher contrast concentrates at specific points in the reasoning trajectory, using a Qwen3-4B code-generation checkpoint ($\alpha = 0.3$). For each rollout, we identify *pivot tokens* as positions containing an explicit self-correction cue – *wait*, *but*, *however*, *alternatively*, *actually*, *let me reconsider*, *let me re-examine*,

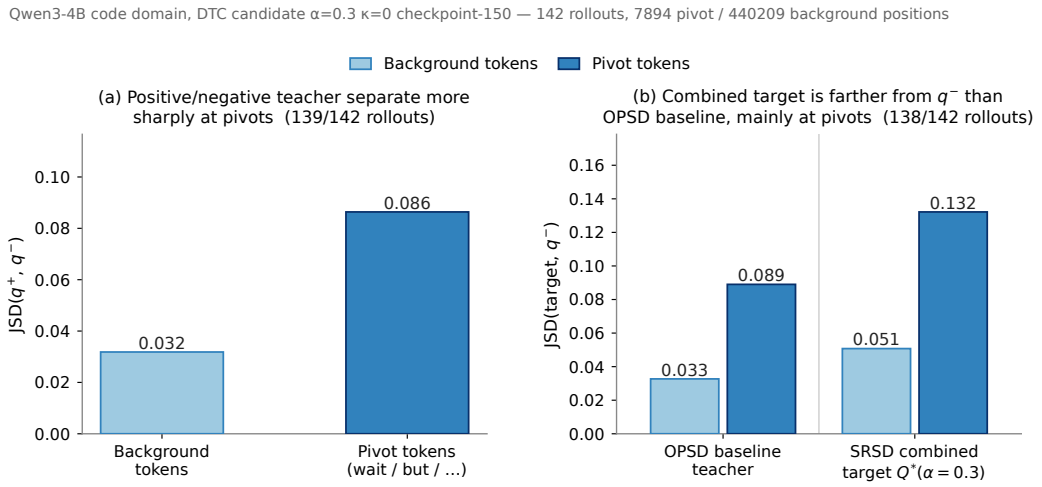

Figure 6: Token-localized teacher divergence at pivot tokens. (a) $\mathrm{JSD}(q^+\|q^-)$ on background versus pivot tokens. (b) $\mathrm{JSD}(\cdot\|q^-)$ for the OPSD baseline teacher and for the SRSD combined target $Q^*_{0.3}$ (Eq. (17)), on background versus pivot tokens. Statistics are computed on a Qwen3-4B code-generation checkpoint ($\alpha = 0.3$), aggregated over 142 rollouts and weighted by token position.

or *hmm* – and treat all remaining generated positions as *background tokens*. Across 142 rollouts containing at least one pivot token, this yields 7,894 pivot positions and 440,209 background positions. We evaluate the same generated tokens under the OPSD baseline teacher (conditioned on the ground-truth solution alone), the SRSD positive teacher q^+, and the SRSD negative teacher q^-, all sharing the same generation and differing only in the conditioning prompt. All divergences are computed over the full vocabulary, without top-k restriction, and aggregated by weighting each token position equally rather than each rollout equally.

Teacher divergence at pivot tokens. We examine two complementary questions. First, whether the two teachers that directly enter $\mathcal{L}_{\mathrm{SRSD}}$ are themselves more distinguishable at pivot tokens. Figure 6(a) shows that $\mathrm{JSD}(q^+\|q^-)$ rises from 0.032 on background tokens to 0.086 on pivot tokens, an increase of about 170% observed consistently in 139 of the 142 rollouts, indicating that the contrastive signal available to SRSD is itself sharper at self-correction points.

Separation from the negative teacher at pivot tokens. Second, we ask whether the SRSD objective induces a target that is farther from the negative teacher than the standard OPSD teacher, and whether this separation is concentrated at pivot tokens. With $\beta = 0$, $\mathcal{L}_{\mathrm{SRSD}} = \mathcal{L}_+ - \alpha\mathcal{L}_-$ consists of forward-KL terms. Using the softmax cross-entropy gradient, the corresponding zero-gradient solution before enforcing the simplex constraint is

$$Q^*_\alpha = \frac{q^+ - \alpha q^-}{1-\alpha}. \tag{17}$$

For analysis, we project this quantity to a valid distribution by clipping negative entries to zero and renormalizing. We use $\alpha = 0.3$, matching the checkpoint in Figure 6.

Figure 6(b) compares its divergence from q^- with that of the OPSD teacher. For the OPSD teacher, $\mathrm{JSD}(\cdot\|q^-)$ increases from 0.033 on background tokens to 0.089 on pivot tokens; for the projected $Q^*_{0.3}$, it increases from 0.051 to 0.132. The resulting separation gap, $\mathrm{JSD}(Q^*_{0.3}\|q^-) - \mathrm{JSD}(q_{\mathrm{OPSD}}\|q^-)$, is 2.4× larger at pivot tokens (+0.043) than at background tokens (+0.018), consistently across 138 of 142 rollouts. This suggests that SRSD induces stronger separation from the negative teacher particularly at self-correction points.

Table 5: Ablation on the conditioning content of the positive teacher's prompt. Each entry reports the average pass@1 over LiveCodeBench v5 and v6. Default ($r + z^+$) is the setting used in the main experiments. Only r removes the State Tracker diagnosis z^+, while retaining the negative teacher and repulsion term, and therefore differs from standalone OPSD. Only z^+ removes the reference solution r. Self-Correct Rollout replaces r with an online self-corrected rollout of the student's own failed attempt and omits z^+; unlike the Self-Model + Self-Correct setting in Table 4, the corrected rollout is provided directly to the positive teacher rather than used to assist State Tracker diagnosis. Best and second-best results within each model are **bolded** and underlined, respectively. The shaded column denotes the default setting.

Model	Baselines			Positive Teacher Input			
	Base	OPSD	LOPD	**Default** $r + z^+$	**Only** r no z^+	**Only** z^+ no r	**Self-Correct Rollout** replaces r, no z^+
Qwen3-4B	44.45	38.88	47.79	**55.72**	52.34	<u>54.59</u>	51.80
Olmo-7B	45.79	44.57	49.61	**55.35**	52.07	<u>52.91</u>	52.16

A.3 Ablation on the Positive Teacher's Conditioning Content

The default positive teacher in Eq. (6) conditions jointly on the verified reference solution r and the State Tracker's diagnosis z^+. We ablate this conditioning content while keeping the rest of the SRSD objective unchanged, including the negative teacher and the repulsion term $-\alpha\mathcal{L}_-$. We consider four variants, evaluated using the same models, dataset, and baselines as Table 4. *Default ($r + z^+$)* is the setting used in the main experiments and corresponds to Eq. (6). *Only r* removes z^+ and conditions the positive teacher solely on the reference solution, $q_t^+ = \pi_\theta(a \mid x, r, y_{<t})$, matching the OPSD-style positive teacher in Eq. (3); the negative teacher and repulsion term remain active, so this variant differs from the standalone OPSD baseline, which has no negative teacher. *Only z^+* removes r and conditions the positive teacher solely on the State Tracker's diagnosis, $q_t^+ = \pi_\theta(a \mid x, z^+, y_{<t})$, without exposing it to the reference solution. *Self-Correct Rollout* replaces r with an online self-corrected rollout $\hat{y}$ obtained by having the student revise its own failed attempt, while omitting z^+, yielding $q_t^+ = \pi_\theta(a \mid x, \hat{y}, y_{<t})$. This differs from the Self-Model + Self-Correct setting in Appendix A.1, where the self-corrected rollout serves only as auxiliary context for producing z^+ and is not shown directly to the positive teacher. Table 5 reports the average pass@1 over LiveCodeBench v5 and v6, isolating the effect of the positive teacher's conditioning while holding the negative teacher and repulsion term fixed.

Contribution of the dual-teacher structure. Only r retains the negative teacher and repulsion term while reducing the positive teacher's conditioning to the same reference-only input used by standalone OPSD. Nevertheless, it reaches 52.34 on Qwen3-4B and 52.07 on Olmo-7B, outperforming standalone OPSD (38.88 and 44.57) by 13.46 and 7.50 points, respectively, and also surpassing LOPD (47.79 and 49.61) on both backbones. Because Only r and standalone OPSD use the same positive-teacher conditioning, these gains isolate the effect of introducing the negative teacher and repulsion term, showing that the dual-teacher structure contributes substantially even without rollout-relative diagnosis.

Self-corrected rollout versus the reference solution. Replacing the reference solution with an online self-corrected rollout while omitting z^+ yields 51.80 on Qwen3-4B and 52.16 on Olmo-7B, close to the 52.34 and 52.07 achieved by Only r. The differences between the two settings are only 0.54 and 0.09 points, respectively, much smaller than their gains over standalone OPSD. Both variants provide the positive teacher with a single solution-like context—either the verified reference or a self-corrected rewrite of the student's failed attempt—without an explicit diagnosis of the current rollout. Their similar performance suggests that, once the dual-teacher structure is retained, replacing the verified reference with a self-corrected solution has relatively little effect compared with providing rollout-relative diagnosis through z^+.

Removing the reference versus removing the diagnosis. Only z^+ reaches 54.59 on Qwen3-4B and 52.91 on Olmo-7B, only 1.13 and 2.44 points below the default (55.72 and 55.35), and is the

strongest of the three ablated variants on both backbones. By comparison, Only r falls 3.38 and 3.28 points below the default, showing a larger degradation when z^+ is removed than when the reference solution is removed. Importantly, the State Tracker constructs z^+ solely from the problem x and the current rollout y, without access to the reference solution. Its stronger performance therefore suggests that explicitly characterizing the current rollout state provides more useful guidance to the positive teacher than conditioning on the reference solution alone. In particular, z^+ identifies the current progress, unresolved issues, and potential errors of the rollout, providing state-specific guidance about how generation should proceed from the current trajectory.

Robustness across conditioning choices. All three ablated variants remain above LOPD on both backbones; even the weakest reaches 51.80 on Qwen3-4B and 52.07 on Olmo-7B, exceeding LOPD by 4.01 and 2.46 points, respectively. Together with the State Tracker and State Guard ablations in Table 3, these results show that SRSD's gains are not tied to a single choice of positive-teacher conditioning. At the same time, the consistently stronger performance of Only z^+ over the other ablated variants highlights rollout-relative diagnosis as the more important component of the positive teacher's context.

A.4 Hyperparameter Ablation on Mathematical Reasoning

We further ablate four training hyperparameters of SRSD on Qwen3-1.7B for mathematical reasoning, starting from the default recipe used to obtain Table 3 ($\alpha = 0.3$, 200 training steps). Each variant perturbs a single hyperparameter while holding the rest at their default value: the JSD mixture coefficient β (Eq. (10)), the learning rate, the rollout temperature, and the max rollout tokens. Table 6 reports Avg@12 on AIME24 (A24), AIME25 (A25), and HMMT25 (H25).

Table 6: Hyperparameter ablation on Qwen3-1.7B for mathematical reasoning. Reference uses the default recipe of Table 3 ($\alpha = 0.3$, 200 training steps). Each other row perturbs a single hyperparameter. We report Avg@12 on AIME24 (A24), AIME25 (A25), and HMMT25 (H25), with **Avg** denoting their mean. Best and second-best results in each column are **bolded** and underlined, respectively. The shaded row denotes the default setting.

Variant	Avg	A24	A25	H25
Reference ($\beta = 0$, lr$= 5 \times 10^{-6}$, $T = 1.1$, len$= 1024$)	**45.65**	**59.44**	45.56	**31.94**
$\beta = 1.0$	40.93	53.61	41.11	28.06
lr $= 3 \times 10^{-6}$	44.44	58.33	**46.11**	28.89
$T = 0.9$	43.15	55.28	44.17	30.00
max completion length $= 2048$	44.07	58.61	44.44	29.17

Increasing β toward the reverse-KL end of the JSD mixture (Eq. (10)) degrades performance the most, lowering Avg by 4.72 points relative to the reference. Lowering the learning rate to 3×10^{-6} slightly improves AIME25 but reduces Avg by 1.21 points, indicating the default learning rate is already close to optimal. Reducing the rollout temperature to 0.9 and doubling the maximum completion length to 2048 both underperform the reference by more than 1.5 points, showing that neither lower-variance rollouts nor longer generations improve on the default 1024-token budget. Overall, the reference configuration used throughout the main results remains the best setting under this sweep.

A.5 α/κ Ablation on Code and Agentic Tool Use

We further extend the ablation of the repulsion weight α and element-level clipping threshold κ from Section 5.3 to code generation and agentic tool use. We jointly sweep $\alpha \in \{0.1, 0.3, 0.6\}$ and $\kappa \in \{0, 0.05\}$. Table 7 reports pass@1 averaged over LiveCodeBench v5 and v6, while Table 8 reports the mean accuracy across BFCL v3 subsets.

The results show that different combinations of α and κ lead to comparable performance across both domains. The best-performing configuration varies across model scales and tasks, suggesting that the optimal balance between the positive and negative teacher signals can depend on the specific

Table 7: Repulsion weight α and clipping threshold κ ablation on coding. We report pass@1 averaged over LiveCodeBench v5 and v6. Best and second-best results for each model are **bolded** and underlined, respectively.

	$\alpha = 0.1$		$\alpha = 0.3$		$\alpha = 0.6$	
Model	$\kappa = 0$	$\kappa = 0.05$	$\kappa = 0$	$\kappa = 0.05$	$\kappa = 0$	$\kappa = 0.05$
Qwen3-4B	53.16	<u>55.41</u>	54.03	54.32	**55.72**	54.59
Qwen3-8B	55.02	55.31	56.23	**57.07**	<u>56.93</u>	55.45
Olmo3-7B	53.75	53.04	54.82	54.13	**55.35**	<u>55.19</u>

Table 8: Repulsion weight α and clipping threshold κ ablation on agentic tool use. We report mean accuracy across BFCL v3 subsets. Best and second-best results for each model are **bolded** and underlined, respectively.

	$\alpha = 0.1$		$\alpha = 0.3$		$\alpha = 0.6$	
Model	$\kappa = 0$	$\kappa = 0.05$	$\kappa = 0$	$\kappa = 0.05$	$\kappa = 0$	$\kappa = 0.05$
Qwen3-4B	<u>26.25</u>	25.38	27.25	26.63	**28.25**	25.87
Qwen3-8B	28.12	26.63	<u>29.50</u>	28.87	**30.38**	28.88

evaluation setting. In all cases, SRSD achieves strong results under the explored hyperparameter range, and we use the corresponding best configuration for each reported setting.

For coding, we additionally evaluate Qwen3-8B in the appendix because directly comparable baseline results at this scale are unavailable. The vanilla model achieves 51.61% and 40.46% pass@1 on LiveCodeBench v5 and v6, respectively, while SRSD improves these results to 60.81% and 53.43%, corresponding to gains of 9.20 and 12.97 points. These results provide additional evidence that SRSD generalizes effectively across model scales and coding benchmarks.

B Data, Training, and Evaluation

Across all three domains, the State Tracker analyzes each supervised on-policy rollout and assigns an error-status marker $e \in \{\mathrm{correct}, \mathrm{wrong}, \mathrm{trunc}\}$ as part of the resulting state representation. When $e = \mathrm{correct}$, the rollout exhibits a viable reasoning structure, and the analysis records the established information and the next subgoal available for verification. When $e = \mathrm{wrong}$, the analysis identifies the local error associated with the observed failure and indicates a possible direction for correction or backtracking. When $e = \mathrm{trunc}$, the rollout terminates before reaching a complete conclusion, so the analysis summarizes only the observed state without making strong assumptions about the unseen continuation.

All three training datasets undergo a single offline filtering pass. For each problem, the base model samples four rollouts, and we discard the problem if all four are correct. Thus, every retained problem has at least one incorrect rollout under the base policy. During training, the negative branch requires an online failed rollout from the current policy. For each question, we sample up to eight rollouts to obtain such a failure. If all eight rollouts are correct, we skip that question for the current training step.

B.1 Mathematical Reasoning

For mathematical reasoning, the State Tracker and State Guard operate as described in the main text. Evaluation follows the setting of the OPSD baseline (Zhao et al., 2026a). We draw 12 samples per problem on AIME24, AIME25, and HMMT25 and report Avg@12, consistent with the results in Table 2.

B.2 CODING

For code generation, the State Tracker and State Guard follow the same construction as in mathematical reasoning. Evaluation follows the LOPD protocol (Zhang et al., 2026). We use the official LiveCodeBench (Jain et al., 2025) code-generation-lite harness and report pass@1 on release v5 and v6. Specifically, we adopt the LOPD-aligned versions, including the 279-task v5 release dated from August 2024 to February 2025 and the 131-task v6 release dated from February to May 2025. We note that LiveCodeBench also provides alternative releases, including a 167-task v5 version and a later 175-task v6 version, which evaluate on different problem sets. To ensure direct comparability with the baseline results, we strictly follow the versions used in the LOPD protocol rather than mixing results across different releases. Generation uses a temperature of 1.0, a presence penalty of 1.5, a repetition penalty of 1.0, and a 48k-token generation limit, matching the settings used to obtain the reported baseline results.

B.3 AGENTIC TOOL USE

Agentic tool use retains the same State Tracker and positive-teacher construction. The State Tracker analyzes the supervised interaction rollout under the original task contract, and the positive teacher conditions on the resulting state representation together with the reference solution r. The negative branch additionally exploits the structured task specification available in tool-use settings. For each training instance, the State Guard corrupts the task contract and tool schema, and provides the resulting mis-specified specification to the negative teacher together with a failed on-policy rollout. The negative teacher therefore observes both an explicit environment-level corruption and a policy-induced failure, exposing malformed, incorrectly parameterized, or otherwise invalid tool-use behaviors that the student should avoid.

Evaluation follows the LOPD protocol (Zhang et al., 2026). We use the official Berkeley Function Calling Leaderboard V3 codebase (Patil et al., 2025) and evaluate on its four multi-turn subsets: base, missing-function, missing-parameter, and long-context. The model is served in function-calling (FC) mode rather than prompted to emit tool calls as text. We report both the individual subset scores and their average, consistent with Table 1.

B.4 TRAINING CONFIGURATION

We use a unified training recipe across all three domains. The student is implemented as a LoRA adapter on the base model and is the only branch that receives gradients. Both teachers use the same base model with the adapter disabled. Consequently, the student and the two teachers share the same underlying base-model weights and differ only in their conditioning information.

Table 9 summarizes the settings that vary across domains, including the training data, rollout lengths, clipping and repulsion strengths, and evaluation protocols. The two sequence-length settings serve different purposes. Max rollout tokens limits only the generated trajectory y at each training step and therefore determines the number of generated positions over which the three branches are scored. Max sequence length instead limits the complete packed sequence, including both the prompt and generation, and must accommodate the longest teacher input. In particular, the positive-teacher prompt additionally contains the reference solution and State Tracker analysis. This accounts for the substantially larger max sequence length in mathematical reasoning, where a 1024-token rollout may be preceded by several thousand tokens of teacher-side conditioning.

C LIMITATIONS

Despite the improvements observed across reasoning, coding, and agentic tasks, SRSD has several limitations. First, the State Tracker introduces additional online analysis cost, as each sampled rollout requires an extra inference step to obtain the rollout-level state representation. Although our experiments show that using the student model itself as the analyzer remains competitive with only small deviations from the default setting, reducing this overhead through more efficient state extraction remains an important direction.

Table 9: Per-domain training and evaluation settings. Rows above the middle rule specify training configurations, while rows below it specify evaluation configurations. Entries marked n/a indicate the setting does not apply to that domain's decoding path.

Setting	Mathematical reasoning	Coding	Agentic tool-use
Training configurations			
Backbones	Qwen3-1.7B / 4B / 8B	Qwen3-4B, Olmo3-7B, Qwen3-8B	Qwen3-4B, Qwen3-8B
Training data	OpenThoughts, 30k math subset	TACO verified split	EnvScaler train split
Source of z^-	failing online rollout	failing online rollout	corrupted contract and failing online rollout
Max rollout tokens	1024	4096	4096 per turn, 32768 over at most 30 turns
Max sequence length	20000	24576	40960
Tracker max new tokens	16384	16384	8192
Rollout temperature	1.1	1.1	0.7
Student thinking mode	Disabled	Disabled	Disabled
β	0	0	0
Clipping rule	Element-level	Element-level	Element-level
κ	0	0	0
α	0.3	0.6	0.6
Training steps	200	200	200
Evaluation configurations			
Benchmarks	AIME24 (30), AIME25 (30), HMMT25 (30)	LiveCodeBench v5 (279), v6 (131)	BFCL v3 (4×200)
Metric	Avg@12	pass@1	per-subset accuracy
Eval. temperature	1.1	1.0	0.7
Eval. max generate tokens	38912	49152	10000 per turn, 40960 over at most 30 turns
Eval. context length	40960	40960	40960
Presence penalty	0	1.5	n/a
Repetition penalty	n/a	1.0	n/a

Second, the quality of state-relative supervision depends on the ability of the analyzer to identify useful reasoning states and repair directions. Our current implementation relies on prompted analysis models and predefined error-status templates, which may not fully capture complex failure modes or domain-specific reasoning patterns. Future work could investigate learned state representations or jointly optimized state trackers.

Third, the negative teacher in SRSD is constructed from failures sampled from the current policy. While this design provides online failure-aware guidance, it only exposes failure modes that are reachable by the current policy distribution and does not guarantee coverage of all possible incorrect behaviors. Exploring broader failure discovery mechanisms may further strengthen the repulsive signal.

Finally, our evaluation focuses on relatively small-to-medium open models and three representative domains. Whether state-relative supervision scales to larger models, longer-horizon tasks, and more diverse interactive environments remains an open question.

D Prompt Templates

This appendix gives the three prompts of one training step as they are issued. All three carry the same generated block y, so the differences below are entirely differences of conditioning. The templates shown are those of mathematical reasoning; the other two domains keep the same three-part organisation, with the rollout replaced by the submitted program or the multi-turn trajectory.

D.1 STUDENT ROLLOUT PROMPT

The student sees the problem and nothing else. The empty think block is inserted by the chat template because the student branch runs with thinking disabled; generation starts immediately after it and produces y.

```
<|im_start|>user
Problem: {problem}

Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>
<|im_start|>assistant
<think>

</think>

{y starts here}
```

D.2 STATE TRACKER REQUEST

Once y is complete it is graded, and the request below returns z^+. The error status e of the rollout selects one of three templates; the wrong-answer one is shown in full. The analysis model reads the problem and the current online rollout, and is asked to identify the rollout's current reasoning state and provide targeted, state-specific guidance.

```
You are analyzing a failed attempt at a problem.
Your diagnosis will be used to provide targeted, state-specific guidance
about the student's current reasoning trajectory.

Problem:
{problem}

Student's incorrect attempt:
{y}


Write a precise diagnostic (4-6 sentences). Cover all four points:
1. LOCATE: Quote the exact phrase, claim, or step from the attempt
   where the reasoning first goes wrong.
2. DIAGNOSE: Name the specific error -- wrong assumption, flawed
   inference, misapplied rule, or incorrect operation.
3. EXPLAIN: State what the student appeared to believe, and why that
   belief is incorrect.
4. CORRECT: Describe the right move at that exact point (one or two
   steps only).

Rules:
- Be specific: name the concept, rule, or principle involved where
  relevant.
- Do not restate or summarize the full solution.
- Do not use <think> tags.
- Write so that someone reading only your diagnosis understands the
  error and the fix without needing the full solution.
```

For $e = \text{correct}$, the request announces a successful attempt and asks for two points, the step worth keeping and the reason it works, with an explicit instruction that alternative valid reasoning paths are acceptable. For $e = \text{trunc}$ it announces an incomplete attempt and asks for four, the goal, the approach taken, the state reached, and the next step, with the missing final answer ruled out as a mistake and the model told not to invent an error. Reusing the wrong-answer template in either case would put a fabricated error into the positive teacher's prompt.

D.3 POSITIVE TEACHER PROMPT

The positive teacher reads the reference solution and z^+, then continues from the same assistant marker with the student's own y appended. It does not see the rollout itself; z^+ is introduced as a diagnosis of an attempt the teacher cannot read, which is why the marker block is framed rather than pasted in bare.

```
<|im_start|>user
Problem: {problem}

Here is a reference solution to this problem:
=== Reference Solution Begin ===
{solution}
=== Reference Solution End ===

A student attempting this problem made a specific mistake. You do
NOT see their attempt -- only this diagnosis of it:
=== Common Pitfall Begin ===
{z+}
=== Common Pitfall End ===
Pay extra attention to the step identified above.

After reading the reference solution above, make sure you truly
understand the reasoning behind each step -- do not copy or
paraphrase it. Now, using your own words and independent reasoning,
derive the same final answer to the problem above. Think step by
step, explore different approaches, and don't be afraid to backtrack
or reconsider if something doesn't work out:

Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>
<|im_start|>assistant
{y}
```

The marker name and the two framing sentences follow e: `Common Pitfall` for a wrong answer, `Reasoning Highlight` for a correct one, and `Partial Progress` for a truncated one, the last worded neutrally on correctness because the assessment it carries may report either. When no z^+ is available the block is dropped rather than emitted empty, which leaves the prompt identical to a plain OPSD teacher prompt.

D.4 NEGATIVE TEACHER PROMPT

The negative teacher receives the same wrapper with one substitution: the failed rollout z^- takes the place of the reference solution, under the same header. It is not told that this text failed, and it is given neither the true solution nor z^+ — either would let it derive the correct continuation, which is what a failure reference must not do. The think markers of z^- are stripped so that the teacher opens its own think block cleanly; the reasoning inside them is kept, since that is the failure the teacher is meant to absorb.

```
<|im_start|>user
Problem: {problem}

Here is a reference solution to this problem:
=== Reference Solution Begin ===
{z-, an independently sampled rollout that fails to reach a correct
 conclusion}
=== Reference Solution End ===

After reading the reference solution above, make sure you truly
understand the reasoning behind each step -- do not copy or
paraphrase it. Now, using your own words and independent reasoning,
derive the same final answer to the problem above. Think step by
```

```
step, explore different approaches, and don't be afraid to backtrack
or reconsider if something doesn't work out:

Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>
<|im_start|>assistant
{y}
```

D.5 THE SUPERVISED TRAJECTORY

Only y is supervised. The three prompts above are each concatenated with the same token block, prompt positions are masked out of the labels, and the three distributions are read at the positions of y, so a token index means the same thing in all three branches.

```
student    : [ student prompt          ][ y ]  -> s_t
teacher(+) : [ positive teacher prompt ][ y ]  -> q_t^+
teacher(-) : [ negative teacher prompt ][ y ]  -> q_t^-
               labels = -100 (masked)    scored positions
```

The three prompts differ in length, so each branch is sliced at its own prompt boundary before the losses are formed. Because y is the student's own sample, the positions being scored are exactly the ones the current policy visits.

E CASE STUDIES

We present two representative AIME24 cases comparing the generations of the OPSD baseline (Zhao et al., 2026a) and SRSD on a Qwen3-1.7B checkpoint. The first case isolates a localized computational error after the two methods follow nearly identical derivations, while the second shows that SRSD can follow a distinct valid derivation even when both methods reach the correct answer. We present key case-study trajectories below to illustrate how OPSD and SRSD differ in their reasoning processes.

E.1 CASE I: LOCALIZED DIVERGENCE AT A CRITICAL ALGEBRAIC STEP

The first case considers AIME24 problem 27. The problem specifies a tetrahedron with $AB = CD = \sqrt{41}$, $AC = BD = \sqrt{80}$, and $BC = AD = \sqrt{89}$, so all four faces are congruent triangles with side lengths $\sqrt{41}$, $\sqrt{80}$, and $\sqrt{89}$. The goal is to determine the common distance from the incenter to each face.

The OPSD baseline and SRSD initially follow the same derivation. Both compute the tetrahedron volume using the Cayley–Menger determinant and obtain $V = 160/3$. Both then use $V = \frac{1}{3}h \cdot$ (sum of the four face areas), reducing the remaining calculation to the area of a single triangular face. Their derivations diverge only at this local subproblem.

OPSD baseline. The OPSD baseline applies Heron's formula directly to the three side lengths:

```
Area = (1/4) * sqrt[(a + b + c)(-a + b + c)(a - b + c)(a + b - c)]

But again, with square roots.

Alternatively, maybe we can compute the square of the area.

Let me denote:

Let me denote:

Let me compute the product:
```

```
(a + b + c)(-a + b + c)(a - b + c)(a + b - c)

This is equal to [(a + b + c)(-a + b + c)(a - b + c)(a + b - c)].

This product is equal to [(b + c)^2 - a^2][(a + c)^2 - b^2][(a +
b)^2 - c^2]... Wait, no, maybe not. Wait, actually, the formula for
the product is known as the product formula in Heron's formula.

Alternatively, perhaps we can use the fact that:

(a + b + c)(-a + b + c)(a - b + c)(a + b - c) = 2(a^2 b^2 + b^2 c^2
+ c^2 a^2 - a^4 - b^4 - c^4)

Wait, but I need to check this.

Alternatively, maybe compute the value step by step.

But given the time constraints, perhaps I need to think of another
way.

Alternatively, maybe we can use the fact that if the tetrahedron has
opposite edges equal, then it's a special type of tetrahedron, and
perhaps we can find the area using vectors or something.

Alternatively, maybe think of the area of the triangle as the same
as the area of another triangle with sides given.

Alternatively, since all four faces are congruent, their areas are
the same, and since the total surface area is 4 * area_of_one_face.

But how to compute area_of_one_face?

Alternatively, maybe compute using the formula for the area of a
triangle with sides a, b, c:

Area = (1/4) * sqrt( (a + b + c)(-a + b + c)(a - b + c)(a + b - c) )

But maybe we can compute this expression numerically?

Let me try.

First, compute the terms:

Let me denote:

a = sqrt(41), b = sqrt(80), c = sqrt(89)

Compute the product:

(a + b + c)(-a + b + c)(a - b + c)(a + b - c)

Let me compute each term:

Let me compute each of the four factors.

Let me denote:

First term: (a + b + c) = sqrt(41) + sqrt(80) + sqrt(89)

Second term: (-a + b + c) = -sqrt(41) + sqrt(80) + sqrt(89)

Third term: (a - b + c) = sqrt(41) - sqrt(80) + sqrt(89)

Fourth term: (a + b - c) = sqrt(41) + sqrt(80) - sqrt(89)
```

```
So the product is:

(sqrt(41) + sqrt(80) + sqrt(89))(-sqrt(41) + sqrt(80) +
sqrt(89))(sqrt(41) - sqrt(80) + sqrt(89))(sqrt(41) + sqrt(80) -
sqrt(89))

This is of the form (x + y + z)(-x + y + z)(x - y + z)(x + y - z),
where x = sqrt(41), y = sqrt(80), z = sqrt(89)

But there's a formula for this. Recall that (x + y + z)(-x + y +
z)(x - y + z)(x + y - z) = 2(a^2 b^2 + b^2 c^2 + c^2 a^2 - a^4 - b^4
- c^4) ?

Wait, actually, I recall that (x + y + z)(-x + y + z)(x - y + z)(x +
y - z) = 2(x^2 y^2 + y^2 z^2 + z^2 x^2) - (x^4 + y^4 + z^4)

But not sure. Alternatively, I can use substitution.

Let me set:

Let me note that:

Let me denote A = sqrt(41), B = sqrt(80), C = sqrt(89).

Then, the product is:

(A + B + C)(-A + B + C)(A - B + C)(A + B - C)

Let me note that this is equal to [(B + C)^2 - A^2][(A + C)^2 -
B^2][(A + B)^2 - C^2]... Wait no, not sure.

Alternatively, maybe compute the product step by step.

Let me compute [(A + B + C)(-A + B + C)] * [(A - B + C)(A + B - C)]

First compute [(A + B + C)(-A + B + C)] = [ (B + C)^2 - A^2 ] = (B^2
+ 2BC + C^2) - A^2

Similarly, [(A - B + C)(A + B - C)] = [A^2 - (B - C)^2] = A^2 - (B^2
- 2BC + C^2) = A^2 - B^2 + 2BC - C^2

Therefore, the entire product is [ (B^2 + 2BC + C^2 - A^2 ) ] * [A^2
- B^2 + 2BC - C^2 ]

Let me denote:

Let me compute the first term: (B^2 + C^2 - A^2) + 2BC

Second term: (A^2 - B^2 - C^2) + 2BC

Wait, no:

Wait, (B + C)^2 - A^2 = B^2 + 2BC + C^2 - A^2

Similarly, (A - B + C)(A + B - C) = A^2 - (B - C)^2 = A^2 - B^2 +
2BC - C^2

Therefore, the entire product is:

[ (B^2 + C^2 - A^2) + 2BC ] * [ (A^2 - B^2 - C^2) + 2BC ]

But this seems complicated. However, if we denote:

Let me note that A^2 = 41, B^2 = 80, C^2 = 89
```

```
So compute each term:

First term:

B^2 + C^2 - A^2 = 80 + 89 - 41 = 128

Then, 2BC: 2 * sqrt(80) * sqrt(89) = 2sqrt(80*89)

Second term:

A^2 - B^2 - C^2 = 41 - 80 - 89 = 41 - 169 = -128

Then, +2BC = same as above, 2sqrt(80*89)

Therefore, the entire product becomes:

[128 + 2sqrt(80*89)] * [-128 + 2sqrt(80*89)]

Let me denote X = 2sqrt(80*89), then the product is (128 + X)(-128 +
X) = X^2 - (128)^2

Therefore:

= X^2 - 16384

Where X = 2sqrt(80*89)

So compute X^2:

X^2 = 4 * (80 * 89) = 4 * 7120 = 28480

Therefore, the entire product is 28480 - 16384 = 12096

Therefore, the product (a + b + c)(-a + b + c)(a - b + c)(a + b - c)
= 12096

Therefore, the area squared is 12096 / 4 = 3024

Wait, no. Wait, Heron's formula says Area^2 = s(s - a)(s - b)(s -
c). But earlier I had:

In the formula above, I calculated [(a + b + c)(-a + b + c)(a - b +
c)(a + b - c)] = 12096, which is equal to 4 * Area^2.

Wait, because Heron's formula says:

Area^2 = s(s - a)(s - b)(s - c)

But in our case, the product (a + b + c)(-a + b + c)(a - b + c)(a +
b - c) equals 4 * Area^2 ?

Wait, let me check.

Wait, no, if we have:

The product [(a + b + c)(-a + b + c)(a - b + c)(a + b - c)] is equal
to 4 * Area^2 ?

Wait, no. Earlier, I used the formula that:

For Heron's formula, the product is 16 * Area^2 ?

Wait, let me recall:

Actually, Heron's formula says:
```

```
Area = sqrt[s(s - a)(s - b)(s - c)]

Therefore, Area^2 = s(s - a)(s - b)(s - c)

But in our case, we computed the product [(a + b + c)(-a + b + c)(a
- b + c)(a + b - c)] which is equal to 4 * Area^2.

But in our case, the product we computed was equal to 12096, which
equals 4 * Area^2 ?

Wait, no, let me see.

Wait, the product (a + b + c)(-a + b + c)(a - b + c)(a + b - c) = 4
* Area^2 ?

Wait, in our case, we computed that product as 12096, which is equal
to 4 * Area^2 ?

If so, then Area^2 = 12096 / 4 = 3024.
```

The baseline correctly evaluates the Heron product as

$$(a+b+c)(-a+b+c)(a-b+c)(a+b-c) = 12096,$$

but then applies the wrong coefficient relating this product to the squared area. Specifically, it divides by 4 rather than 16, producing $\text{Area}^2 = 3024$ instead of the correct value 756, and hence $12\sqrt{21}$ rather than $6\sqrt{21}$. Notably, the rollout repeatedly revisits the relevant identity (e.g., "Wait, no..."), yet fails to resolve the coefficient error. The subsequent computation is otherwise consistent, so this local error propagates directly to the final answer.

SRSD. SRSD reaches the same face-area subproblem but takes a different local direction. Rather than evaluating the Heron product, it computes the included angle using the Law of Cosines and then applies $\frac{1}{2}ab\sin\theta$:

```
For triangle ABC, with sides AB, AC, and BC. Let me denote AB = c,
AC = b, BC = a. Wait, standard notation is usually a, b, c for
opposite angles. But maybe this is getting too confusing.

Alternatively, let me use the formula:

Area = (1/4) * sqrt( (a + b + c)(-a + b + c)(a - b + c)(a + b - c) )

But in this case, since the sides are sqrt(41), sqrt(80), sqrt(89),
let me denote them as:

Let's let's say the sides are:

a = sqrt(41), b = sqrt(80), c = sqrt(89)

Then, the semi-perimeter s = (a + b + c)/2.

Then, the area is sqrt[ s (s - a)(s - b)(s - c) ]

But this seems messy with square roots. However, maybe we can
compute the squares of the areas?

Alternatively, perhaps I can use the formula:

Area = (1/2)ab sin(theta), where theta is the angle between sides a
and b.

But if I can find theta, then compute the area.
```

```
But again, that would require using the Law of Cosines.

Let me try that.

For triangle ABC:

Let's denote AB = c = sqrt(41), AC = b = sqrt(80), BC = a =
sqrt(89).

Then, by the Law of Cosines:

BC^2 = AB^2 + AC^2 - 2 AB * AC cos(theta), where theta is the angle
at A.

Wait, BC^2 = AB^2 + AC^2 - 2 AB * AC cos(angle BAC).

Therefore:

89 = 41 + 80 - 2 * sqrt(41) * sqrt(80) * cos(theta)

So:

89 = 121 - 2 * sqrt(41*80) * cos(theta)

Therefore:

2 * sqrt(41*80) * cos(theta) = 121 - 89 = 32

So:

cos(theta) = 32 / [2 * sqrt(41 * 80)] = 16 / sqrt(41 * 80)

Compute sqrt(41*80):

sqrt(41*80) = sqrt(41*8*10) = sqrt(328*10) = sqrt(3280). Wait, but
maybe I can compute it numerically? Wait, but perhaps I can
rationalize it.

Wait, let's compute:

cos(theta) = 16 / sqrt(41*80) = 16 / sqrt(3280). Let me see if 3280
can be simplified.

But maybe not necessary yet. The area of triangle ABC is (1/2) * AB
* AC * sin(theta). So once I compute sin(theta), I can compute the
area.

But to compute sin(theta), I can use sin^2(theta) + cos^2(theta) =
1.

So, sin(theta) = sqrt(1 - cos^2(theta)).

Therefore, sin(theta) = sqrt(1 - [256 / 3280]).

Wait, compute 256 / 3280:

Divide numerator and denominator by 16: 16 / 205. So 16/205. Then, 1
- 16/205 = (205 - 16)/205 = 189 / 205.

Therefore, sin(theta) = sqrt(189/205) = sqrt(189)/sqrt(205).
Simplify sqrt(189) = sqrt(9*21) = 3*sqrt(21). Therefore, sin(theta)
= 3*sqrt(21)/sqrt(205).

Thus, the area of triangle ABC is (1/2)*AB*AC*sin(theta) =
```

```
(1/2)*sqrt(41)*sqrt(80)*(3*sqrt(21)/sqrt(205)).

Simplify:

sqrt(41)*sqrt(80) = sqrt(41*80) = sqrt(3280).

Therefore, Area ABC = (1/2) * sqrt(3280) * 3 * sqrt(21) / sqrt(205).

Note that sqrt(3280) / sqrt(205) = sqrt(3280 / 205) = sqrt(16) = 4,
because 205*16 = 3280. Let me check:

205 * 16 = 205*10 + 205*6 = 2050 + 1230 = 3280. Yes!

Therefore, sqrt(3280)/sqrt(205) = sqrt(16) = 4.

Therefore, Area ABC = (1/2)*3*4*sqrt(21) = (6)*sqrt(21).

So, the area of triangle ABC is 6sqrt(21).
```

Case summary. SRSD obtains the correct face area, $\text{Area} = 6\sqrt{21}$, by taking a different local route at the point where the baseline fails. The remainder of the two solutions is largely shared: both compute the same tetrahedron volume, reduce the problem to the same face-area calculation, and use the same final volume–area relation. The difference is therefore localized to a specific intermediate step rather than to the overall solution strategy. In this case, the alternative continuation taken by SRSD avoids the coefficient error in the baseline's Heron-formula calculation and leads to the correct final result.

E.2 Case II: Distinct Valid Derivations from a Shared Starting Point

The second case considers AIME24 problem 8: given $\log_x(y^x) = \log_y(x^{4y}) = 10$ for $x, y > 1$, determine xy. Unlike Case I, both the OPSD baseline and SRSD reach the correct answer, $xy = 25$. They also rely on the same underlying logarithmic identities and eventually introduce the same substitution, $a = \log_x(y)$, with $\log_y(x) = 1/a$. The difference lies in how they develop the derivation from this common starting point.

OPSD baseline. The OPSD baseline applies the logarithmic power rule directly to the two equations, expresses x and y in terms of the substitution variable, and then eliminates that variable when computing their product:

```
Okay, let me try to figure out this problem. So, we have two
logarithmic equations: log base x of (y^x) equals 10, and log base y
of (x^{4y}) also equals 10. Both x and y are greater than 1. We need
to find the value of xy. Hmm, let me start by recalling some
logarithm properties.

First, I remember that log_b(a) = c means that b^c = a. So, maybe I
can rewrite the given equations using this property. Let me write
down the first equation: log_x(y^x) = 10. If I rewrite that using
the logarithm power rule, which says log_b(a^c) = c*log_b(a), then
this becomes:

log_x(y^x) = x * log_x(y) = 10.

Similarly, the second equation is log_y(x^{4y}) = 10. Applying the
same power rule here: log_y(x^{4y}) = 4y * log_y(x) = 10.

So now we have two equations:

1) x * log_x(y) = 10
2) 4y * log_y(x) = 10
```

```
Hmm, okay. Let me think. Let me denote log_x(y) as some variable.
Let me call it, say, k. Wait, but maybe there's a relationship
between log_x(y) and log_y(x). Since log_b(a) and log_a(b) are
reciprocals of each other, right? Because log_b(a) = 1 / log_a(b).
So, if I let log_x(y) = k, then log_y(x) = 1/k. Let me verify that.

Yes, because if log_x(y) = k, then by definition, x^k = y. Taking
log base y of both sides, log_y(x^k) = log_y(y) => k * log_y(x) = 1
=> log_y(x) = 1/k. So that's correct.

Therefore, if log_x(y) = k, then log_y(x) = 1/k.

So, going back to our equations. The first equation is x * k = 10.
The second equation is 4y * (1/k) = 10.

So, we have two equations:

1) x * k = 10
2) 4y * (1/k) = 10

So maybe we can express both equations in terms of k and relate
them?

But we also know that k = log_x(y). Let me think. Also, since k =
log_x(y), then by definition, y = x^k. Therefore, we can express y
in terms of x and k. Similarly, from equation 1, x * k = 10, so x =
10 / k. Therefore, substituting into y = x^k, we get y = (10 / k)^k.
So, y = 10^k / k^k.

Alternatively, maybe we can find another relationship. Let me check
the second equation. The second equation is 4y / k = 10. Therefore,
4y = 10k => y = (10k)/4 = (5k)/2.

So from equation 2, we have y = (5k)/2. And from equation 1, x = 10
/ k.

[... substitutes y = x^k, sets up (10/k)^k = 5k/2, then
tries several integer values of k (relabeled m in the rollout)
before returning to the algebraic route below...]

Wait, another thought. Let me recall that x and y are both greater
than 1, so maybe we can find relations between x and y?

From x = 10/m and y = (5m)/2, so:

xy = (10/m) * (5m)/2 = (10 * 5m) / (2m) ) = (50m)/(2m) = 25.

Wait, that's interesting! So xy is 25?

Wait, hold on. If xy is equal to 25 regardless of m? Let me check
that again.

From x = 10/m and y = (5m)/2, so multiplying them:

x * y = (10/m) * (5m)/2 = (10 * 5m) / (m * 2) ) = (50m) / (2m) ) =
25. Yes! Because the m cancels out.

So regardless of the value of m, as long as x and y satisfy those
equations, xy = 25.
```

Once $x = 10/k$ and $y = 5k/2$ are established, the substitution variable cancels immediately in the product, yielding $xy = 25$. The baseline therefore reaches the correct answer through a direct algebraic elimination of the shared logarithmic substitution.

SRSD. SRSD takes a different route. It first converts the two logarithmic equations to exponential form, applies natural logarithms, and then relates the resulting equations through $\ln x$ and $\ln y$:

```
Since we know that log_x(y^x) = 10, which is equivalent to y^x =
x^{10}. Because if log_b(a) = c, then a = b^c.

Therefore, from the first equation, log_x(y^x) = 10 implies y^x =
x^{10}.

Similarly, the second equation is log_y(x^{4y}) = 10 implies x^{4y}
= y^{10}.

So we have:

y^x = x^{10} ... (1)

x^{4y} = y^{10} ... (2)

So now we have two equations:

(1) y^x = x^{10}

(2) x^{4y} = y^{10}

So maybe we can take the ratio of equations (1) and (2), or some
other manipulation?

Alternatively, take logarithms of both sides. Let me try that.

Take natural logarithm on both sides of equation (1):

ln(y^x) = ln(x^{10})

Which is x ln y = 10 ln x.

Similarly, for equation (2):

ln(x^{4y}) = ln(y^{10})

Which is 4y ln x = 10 ln y.

So now we have:

From equation (1): x ln y = 10 ln x ... (A)

From equation (2): 4y ln x = 10 ln y ... (B)

So now equations (A) and (B). Let me write them again:

(A): x ln y = 10 ln x

(B): 4y ln x = 10 ln y

Hmm, so perhaps I can relate equations (A) and (B). Let me see. Let
me note that in equation (A), x ln y = 10 ln x. If I divide both
sides by ln x, I get:

x (ln y / ln x) = 10

But ln y / ln x is log_x y. Which is the same as a, which we had
earlier. Which is equal to 10/x (from equation 1). Wait, but earlier
we had a = 10/x, and here, we have:

x * (ln y / ln x) = 10. But ln y / ln x is log_x y = a. So:
```

```
x * a = 10, which is the same as equation (1). So that checks out.
So perhaps not helpful.

But since we have equations (A) and (B), maybe we can express ln y
in terms of ln x or vice versa?

From equation (A): x ln y = 10 ln x => ln y = (10 / x) ln x.

Similarly, from equation (B): 4y ln x = 10 ln y => ln y = (4y / 10)
ln x = (2y / 5) ln x.

So now we have two expressions for ln y:

From (A): ln y = (10/x) ln x

From (B): ln y = (2y / 5) ln x

Therefore, equate them:

(10/x) ln x = (2y / 5) ln x

Assuming ln x != 0 (since x > 1, so ln x > 0), we can divide both
sides by ln x:

10/x = 2y / 5

Multiply both sides by 5:

(10/x)*5 = 2y => 50 / x = 2y => y = 25 / x

Which is the same as before. Therefore, y = 25 / x, which gives us
that xy = 25. So this seems consistent. Therefore, according to
this, the answer is 25.
```

Case summary. Both methods solve this problem correctly, but they do not reproduce the same derivation. The OPSD baseline works directly with the logarithmic power rule and eliminates the substitution variable, whereas SRSD first passes through the equivalent exponential and natural-logarithm forms before deriving $y = 25/x$. The two paths ultimately recover the same underlying relation and the same answer, $xy = 25$.

This example complements Case I. The first case shows a localized divergence at an intermediate step where the baseline makes an error and SRSD follows a different valid continuation. Here, no correction is required: both solutions are valid, yet SRSD still follows a distinct derivation rather than reproducing the baseline's particular expansion order. Together, the two cases illustrate that the observed differences are not restricted to trajectories where the baseline fails; state-relative supervision can also preserve alternative valid solution paths when multiple derivations lead to the same answer.