DOES SELF-DISTILLATION IMPROVE REASONING?

Anonymous authors
Paper under double-blind review

ABSTRACT

In self-distillation, a teacher is constructed by prompting a student model with privileged information or environmental feedback, and logit-level feedback is derived from the difference between the two models' output distributions. This paradigm has recently gained traction as an alternative to policy-gradient reinforcement learning, promising dense supervision where reward signals are sparse and resting on the assumption that the self-conditioned teacher approximates the optimal KL-regularized policy. This paper critically evaluates whether self-distillation methods indeed extract useful guidance from privileged information in reasoning tasks or whether their gains are driven by simpler mechanisms. We report four findings: (1) Self-Distillation Policy Optimization (SDPO) only improves performance on LiveCodeBench v6 problems seen during training whereas policy-gradient training on the same reward generalizes to held-out problems; (2) SDPO's reported gains in science reasoning and tool use are largely recoverable through supervised fine-tuning on final answers without chain-of-thought; (3) the gains of On-Policy Self-Distillation (OPSD) in math can be attributed to distilling thinking-mode behavior into the non-thinking student, a mechanism that does not require privileged information; and (4) SDPO and OPSD gradients are not aligned with the policy gradient, contradicting the optimal-teacher assumption. We evaluate across Qwen3 models from 1.7B to 14B and Olmo-3-7B models. Together, these results indicate a need for robust methods that leverage privileged information to improve reasoning performance. Our code is available at `https://anonymous.4open.science/r/does-sd-improve-reasoning/`.

1 INTRODUCTION

Reinforcement learning from verifiable rewards (RLVR) has become a standard approach for finetuning large language models (LLMs) on reasoning tasks such as mathematics and coding (Lambert et al., 2025; Guo et al., 2025). As responses in these domains are automatically verifiable, scaling RLVR is straightforward. However, typical RLVR pipelines use binary rewards that indicate only whether the final answer is correct or whether all unit tests pass. Such rewards provide no information on how attempted solutions should be revised. Process reward models score individual reasoning steps (Wang et al., 2024; Cui et al., 2025), but require training an additional model.

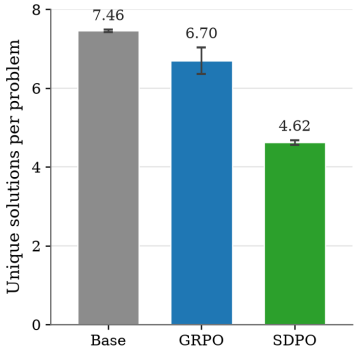


Figure 1: **SDPO reduces solution diversity on LiveCodeBench v6.** AST-unique solutions per training problem (from 8 samples) for Qwen3-14B before (Base) and after 200 steps of GRPO or SDPO (mean ± std, four seeds).

Another way to obtain richer supervision is *knowledge distillation*: when a stronger teacher model is available during training, its capabilities can be transferred into a student model by aligning the logit distributions between the two models (Hinton et al., 2015). When this loss is computed along trajectories sampled from the student, the resulting *on-policy distillation* (OPD) objective provides a dense, per-token learning signal. This bypasses the sparse scalar rewards of RLVR and yields better sample efficiency (Gu et al., 2023; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). OPD has proven effective for training performant small models: the Qwen3 models from 1.7B to 14B parameters were post-trained entirely via distillation rather than RL (Qwen Team, 2025).

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 2: **The reference solution barely changes the self-distillation teacher.** Teacher log-probabilities (thinking on) along a Qwen3-8B thinking-off student rollout on an OpenThoughts problem, with the problem's reference solution in context (ref z, top) and with no solution (no z, bottom). In each panel, the first row is the sampled token and the three rows below are the student's next three most likely tokens. The two teachers assign very similar probabilities, suggesting that OPSD distillation effects do not significantly depend on z (Section 4.3).

Toward the goal of recursive self-improvement (Favaro & Clark, 2026), it is of interest whether a model can improve its capabilities indefinitely from environmental feedback rather than being limited by a fixed external source of supervision. OPD, however, requires access to a stronger external teacher, whose capabilities then bound what the student can learn by imitation. Self-distillation removes this dependence, resting on the hypothesis that *the student itself can act as a stronger teacher when conditioned on additional context available during training*. In reasoning tasks, such context is readily available in the form of feedback far richer than the binary outcome that RLVR consumes, such as reference solutions, critiques, error messages, or failing unit tests. Concretely, in self-distillation the model samples a trace along which it is trained to match the token-level distributions of a (teacher) copy of itself that additionally sees this privileged information (Hübotter et al., 2026; Shenfeld et al., 2026; Zhao et al., 2026a). The self-conditioned teacher therefore converts this rich but unstructured feedback into a dense, per-token signal, without requiring an external model.

However, prior work questions the reliability of self-distillation objectives. The self-conditioned teacher's outputs do not express uncertainty through tokens such as "wait", "hmmm", and "perhaps" and matching the teacher suppresses error-correcting behavior and shortens the student's traces in ways that hurt performance (Kim et al., 2026; Pan et al., 2026; Kaur et al., 2026). The self-conditioned teacher is unable to consistently distinguish between correct and incorrect student generations (Harne et al., 2026; Nguyen et al., 2026). Together, these results raise a more fundamental question: *does self-distillation extract useful guidance from privileged information, or are its reported gains driven by simpler mechanisms?*

We revisit this question through careful reproductions of self-distillation policy optimization (SDPO) (Hübotter et al., 2026) and on-policy self-distillation (OPSD) (Zhao et al., 2026a). We leverage Qwen3 (Qwen Team, 2025) and Olmo-3 (Olmo Team, 2025) models to ensure our observations hold over a range of model sizes and families. We find that the reported gains from privileged-information self-distillation in reasoning domains are weaker than they appear:

- **SDPO's coding gains do not transfer to unseen problems** (Section 4.1). SDPO's LiveCodeBench v6 (Jain et al., 2024) results train and evaluate on the same set of coding problems. Correcting for this reveals that SDPO overfits to its training problems, reducing the diversity of its solutions (Figure 1), and degrades on held-out problems while policy-gradient training on the same verifiable reward generalizes.
- **SDPO's gains in science reasoning and tool use do not require reasoning** (Section 4.2). Supervised finetuning on final answers with *no chain-of-thought* outperforms SDPO in four of five science-reasoning (Feng et al., 2024) and tool-use (Tang et al., 2023) domains.

- **OPSD transfers thinking-mode behavior, not privileged information** (Section 4.3). In math reasoning, privileged information barely changes the self-conditioned teacher's token distribution (Figure 2), and distilling thinking-mode behavior has a substantially larger effect than privileged information within OPSD's reported results.

- **Self-distillation gradients do not align with policy gradients** (Section 4.4). We measure how well each method's training gradient aligns with the policy gradient on held-out problems. On Qwen3 models, OPSD and SDPO gradients are consistently less aligned with this held-out policy gradient than GRPO gradients on the same training problems and often point away from it, violating an assumption of self-distillation methods (Section 3.1).

2 Related works

Reinforcement learning from verifiable rewards (RLVR). The goal of reinforcement learning is to maximize the expectation of reward under a policy's responses. For verifiable tasks in math reasoning and coding, such rewards are assigned by verifying the equivalence between expressions or checking whether a proposed program passes all test-cases. RLVR for LLMs is typically done using policy gradient algorithms (Williams, 1992; Sutton et al., 2000). Although PPO (Schulman et al., 2017) was used in foundational work (Ouyang et al., 2022; Bai et al., 2022), critic-free methods such as GRPO (Shao et al., 2024) and RLOO (Kool et al., 2019; Ahmadian et al., 2024) have become popular due to their simplicity, despite feedback only being given at the trajectory level.

On-policy distillation (OPD). Unlike classic distillation, which tries to match teacher-generated outputs (Hinton et al., 2015), on-policy distillation seeks to boost performance by minimizing per-token divergences between student logits $\pi_\theta(\cdot|x, y_{<t})$ and teacher logits $\pi_T(\cdot|x, y_{<t})$ along prefixes sampled from the *student policy* (Lu & Thinking Machines Lab, 2025; Xu et al., 2026; Gu et al., 2023). OPD has been shown to be effective in post-training small-models (Qwen Team, 2025) and merging the capabilities of multiple domain-specialist expert models (Xu et al., 2026; Kimi Team, 2026; Ma et al., 2026).

Self-distillation and privileged information. Instead of relying on an external stronger model, self-distillation (Shenfeld et al., 2026; Zhao et al., 2026a; Hübotter et al., 2026) parameterizes the teacher distribution as $\pi_T(\cdot|x, y_{<t}) = \pi_\theta(\cdot|x, z, y_{<t})$, with privileged information z which is derived from a reference solution or environmental feedback. This purportedly moves the student towards a higher-reward policy, transferring the reference solution's reasoning content.

Limitations of self-distillation. Prior work documents several failure modes of self-distillation: SDPO generalizes poorly out of distribution and in continual learning settings (Wang et al., 2026), self-distillation degrades thinking models by suppressing backtracking behavior and collapsing response lengths (Kaur et al., 2026; Harne et al., 2026), and the self-conditioned teacher fails to distinguish correct from incorrect student solutions (Nguyen et al., 2026). Concurrently, Shrestha & Tessier (2026) show that OPSD's gains stem from thinking-mode transfer rather than privileged information. Methods that retain GRPO and use the teacher only to reweight advantages such as Pan et al. (2026) fall outside our scope, as we study the self-distillation objective in isolation. We provide an extended discussion of existing work on the limitations of self-distillation, and of how our paper differs from each, in Appendix A.

3 Preliminaries

We introduce the baseline training objectives, then show in Section 3.1 that under an optimal teacher all three share the same update direction, which motivates the gradient analysis of Section 4.4.

Notation. Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ denote the spaces of problems, solutions, and privileged information respectively. We assume all three are finite, as is the case for language models with a finite vocabulary and a maximum sequence length. For $y \in \mathcal{Y}$, we write y_t for its t-th token, $y_{<t}$ for its first $t-1$ tokens, and $|y|$ for its length. For distributions p, q over $\mathcal{Y}$, $D_{\mathrm{KL}}(p \,\|\, q) = \sum_{y \in \mathcal{Y}} p(y) \log \frac{p(y)}{q(y)}$.

Supervised fine-tuning (SFT). Given a distribution over problems $x \sim \rho(\cdot)$ and a distribution over solutions $y \sim \pi_T(\cdot|x)$, supervised finetuning optimizes $\pi_\theta(\cdot|x)$ to maximize the likelihood of y:

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{x\sim\rho(\cdot),y\sim\pi_T(y|x)}\left[\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid x, y_{<t})\right]. \tag{1}$$

Note that optimizing the objective above is equivalent to minimizing D_{KL} from the data distribution to the policy, namely, $\mathbb{E}_{x\sim\rho(\cdot)}[D_{\mathrm{KL}}(\pi_T(\cdot|x) \,\|\, \pi_\theta(\cdot|x))]$.

Group Relative Policy Optimization (GRPO). Given a distribution over problems $x \sim \rho(\cdot)$ and a reward function $R : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, for a given x, GRPO samples a group of K rollouts $y_1, \ldots, y_K \sim \pi_\theta(\cdot|x)$ from the student and computes their rewards $r_i = R(x, y_i)$. The loss is then computed as

$$\mathcal{L}_{\mathrm{GRPO}}(x) = -\frac{1}{\sum_{i=1}^{K}|y_i|}\sum_{i=1}^{K}\sum_{t=1}^{|y_i|}\min\left(w_{i,t}(\theta)\,\hat{A}_i,\ \mathrm{clip}\left(w_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_i\right), \tag{2}$$

where $w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(y_{i,t}|x,y_{i,<t})}$ is the per-token importance ratio and $\hat{A}_i = \frac{r_i - \bar{r}}{\mathrm{std}(r)}$ is a sequence-level z-score normalized advantage shared by all tokens in rollout i. GRPO is a practical instantiation of policy-gradients which aims to maximize $\mathbb{E}_{x\sim\rho(\cdot),y\sim\pi_\theta(\cdot|x)}[R(x,y)]$ by using the following gradient update $\mathbb{E}_{x\sim\rho(\cdot),y\sim\pi_\theta(\cdot|x)}[R(x,y)\nabla_\theta \log \pi_\theta(y|x)]$.

Privileged information self-distillation. Given a distribution over problems $x \sim \rho(\cdot)$ and a distribution over privileged information $z \sim \gamma(\cdot|x)$, we construct a privileged teacher $\pi_T(\cdot|x) = \pi_\theta(\cdot|x, z)$ for problems x for some sampled z. Zhao et al. (2026a) use a fixed reference solution in the dataset as z, while Hübotter et al. (2026) use a successful sibling rollout and environmental feedback if available. The loss is then computed as

$$\mathcal{L}_{\mathrm{SD}} = \mathbb{E}_{x\sim\rho(\cdot),y\sim\mathrm{sg}(\pi_\theta(\cdot|x)),z\sim\gamma(\cdot|x)}\left[\sum_{t=1}^{|y|} D\left(\pi_\theta(\cdot \mid x, y_{<t}), \mathrm{sg}(\pi_\theta(\cdot \mid x, y_{<t}, z))\right)\right], \tag{3}$$

where sg is the stop-grad operator and D represents an information divergence such as forward KL, reverse KL, or Jensen-Shannon divergence. The sampler $y \sim \pi_\theta(\cdot|x)$ is typically not differentiated through due to stability reasons (Agarwal et al., 2024; Hübotter et al., 2026). Common tweaks include an EMA-teacher (Hübotter et al., 2026; Shenfeld et al., 2026), top-k logit approximations (Hübotter et al., 2026), and per-token clipping (Zhao et al., 2026a). Notably, by choosing $D(p, q) = D_{\mathrm{KL}}(p \,\|\, q)$, we have that $\mathcal{L}_{\mathrm{SD}}$ is *equal in value* to $\mathbb{E}_{x\sim\rho(\cdot),\, z\sim\gamma(\cdot|x)}\left[D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_T(\cdot \mid x, z))\right]$.

3.1 Equivalence between Policy Gradient, SFT, and Distillation

Policy gradient improves the policy by ascending expected reward, whereas self-distillation moves the student towards a teacher, so whether it improves depends on the teacher. Self-distillation implicitly assumes that the self-conditioned teacher approximates the KL-regularized optimal policy relative to the current policy, $\pi^*(y|x) \propto \pi_\theta(y|x)\exp(R(x,y)/\beta)$, which we call the optimal teacher (Shenfeld et al., 2026). We show that for this teacher, the policy gradient, SFT on samples from π^*, and on-policy distillation towards π^* share the same update direction. Hence, if the assumption holds, self-distillation gradients should align with the policy gradient, which we test in Section 4.4.

Proposition 3.1. *Consider the single-prompt setting, with a fixed $x \sim \rho(\cdot)$. Fix any policy $\pi_\theta(\cdot|x)$ over $\mathcal{Y}$, a binary reward function $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, and $\beta > 0$, and define the optimal teacher $\pi^*(y|x) = \frac{1}{Z(x)}\pi_\theta(y|x)\exp\left(R(x,y)/\beta\right)$, which maximizes $\mathbb{E}_{y\sim\pi(\cdot|x)}[R(x,y)] - \beta D_{\mathrm{KL}}(\pi(\cdot|x) \,\|\, \pi_\theta(\cdot|x))$ over policies π. Since x is fixed, we suppress it from the notation, writing $\pi_\theta(y)$, $R(y)$, Z, and $\pi^*(y) = \frac{1}{Z}\pi_\theta(y)\exp\left(R(y)/\beta\right)$. Then, we have that:*

$$\frac{\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]}{\|\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]\|_2} = -\frac{\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}\left(\pi^*\right) \,\|\, \pi_\theta)}{\|\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}\left(\pi^*\right) \,\|\, \pi_\theta)\|_2} = -\frac{\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \mathrm{sg}\left(\pi^*\right))}{\|\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \mathrm{sg}\left(\pi^*\right))\|_2}, \tag{4}$$

whenever $\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]$ is non-zero.

We defer the proof to Appendix B.

Crucially, the self-conditioned teacher must stay close to the student. RL generalizes better than SFT on reasoning tasks (Chu et al., 2025), plausibly since on-policy RL induces a smaller KL shift from the base model (Shenfeld et al., 2025); a teacher far from π_θ forfeits this small-KL-shift property.

4 EXPERIMENTAL RESULTS

We now examine whether the reported gains of prior self-distillation works reflect the generalization performance of policy gradients (Section 4.1), require reasoning behavior (Section 4.2), transfer via privileged information (Section 4.3), and have gradients that align with the policy gradient as Proposition 3.1 would require of an optimal teacher (Section 4.4).

4.1 SDPO DOES NOT GENERALIZE TO NEW CODING PROBLEMS

Harne et al. (2026) suggest that the self-conditioned teacher is heavily biased towards the reference solution given in context in a way that does not generalize, which makes the coding gains reported by SDPO (Hübotter et al., 2026) surprising.

SDPO builds its teacher from privileged information available only at training time. On coding tasks, the teacher sees a successful student rollout from the same batch along with environment feedback on execution errors on test cases. Using Qwen3-8B, they claim a 7.6% accuracy improvement on LiveCodeBench v6 (Jain et al., 2024) evaluations compared to GRPO baselines. To reconcile these differences, we show that SDPO's coding evaluation reuses its training problems, and therefore does not measure generalization to new problems.

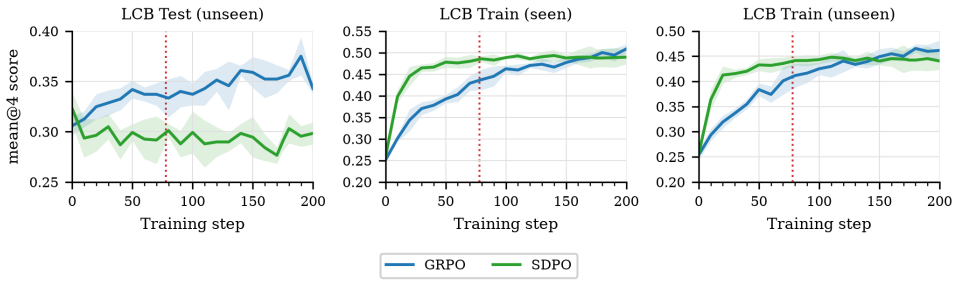


Figure 3: Qwen3-8B SDPO vs. GRPO on LiveCodeBench v6 across four seeds, scored as the fraction of problems with no failing test cases (mean ± std). TRAIN_SEEN and TRAIN_UNSEEN evaluate training problems on seen and on held-out test cases respectively; TEST_UNSEEN evaluates held-out problems. The 20000 training generations reported by Hübotter et al. (2026) is marked in red.

Setup. LiveCodeBench v6 pairs each competitive programming problem with a set of test cases. Concretely, SDPO trains on *all* problems with half of each problem's test cases removed, and evaluates on the unmodified dataset. To separate memorization from generalization, we instead split the problems into disjoint, equal-size sets $\mathcal{P}_{\text{train}}$ and $\mathcal{P}_{\text{test}}$, giving three evaluation groups: TRAIN_SEEN (test cases used during training), TRAIN_UNSEEN (held-out test cases of problems in $\mathcal{P}_{\text{train}}$), and TEST_UNSEEN (all test cases of problems in $\mathcal{P}_{\text{test}}$).

Results. We train Qwen3-8B, Qwen3-14B, and Olmo-3-7B-Instruct on this split with SDPO and GRPO for 200 steps, four seeds each, otherwise following the setup of Hübotter et al. (2026); we provide our hyperparameters in Appendix E.1. Figure 3 shows Qwen3-8B; results for the other two models and tables of accuracies are in Appendix D.1.

SDPO does not improve on new problems. Over 200 steps, GRPO improves TEST_UNSEEN by 3.7, 4.9, and 2.6 points on Qwen3-8B, Qwen3-14B, and Olmo-3-7B-Instruct, whereas SDPO changes it by -2.5, -0.5, and -9.9 points. The disparity is largest on Olmo-3-7B-Instruct, where SDPO ends 13.1 points below GRPO (21.0 vs. 34.1). In summary, we find that mimicking a privileged teacher does not generalize to new problems.

SDPO does generalize to held-out test cases. SDPO's gains on TRAIN_SEEN largely carry over to TRAIN_UNSEEN ($+23.1$ vs. $+18.4$ on Qwen3-8B, $+22.5$ vs. $+17.2$ on Qwen3-14B, and $+18.3$ vs. $+16.7$ on Olmo-3-7B-Instruct), retaining 76–91% of the gain compared with 81–103% for GRPO. SDPO therefore overfits at the level of training problems instead of at the level of unit tests, and produces fewer distinct solutions per training problem than GRPO (Figure 1). We hypothesize this is because a problem's test cases check the same specification, so a program that passes the seen test cases is usually correct and passes the held-out ones too. This reconciles our results with those of

Hübotter et al. (2026); indeed, evaluating on TRAIN_SEEN and TRAIN_UNSEEN at 80 steps across all three models, we find that SDPO beats GRPO.

On Qwen3 models, GRPO overtakes SDPO on training problems with longer training. SDPO improves faster early on but then slows, while GRPO keeps improving and overtakes it after $\sim$160 steps on both TRAIN_SEEN and TRAIN_UNSEEN. At 200 steps on Qwen3-8B, GRPO scores 50.9 on TRAIN_SEEN and 46.2 on TRAIN_UNSEEN, versus 48.9 and 44.0 for SDPO; on Qwen3-14B, GRPO scores 54.1 and 49.2, versus 51.9 and 45.4. The $\sim$80-step comparison of Hübotter et al. (2026) therefore does not reflect performance at convergence—at 80 steps, GRPO is still improving and SDPO's advantage disappears with further training. In Olmo-3-7B-Instruct, SDPO continues to outperform GRPO on training problems.

4.2 SDPO DOES NOT BEAT SFT ON NON-REASONING TASKS

Given that SDPO overfits on coding problems, we also examine its reported gains in science reasoning and tool use, where Hübotter et al. (2026) find that SDPO outperforms GRPO and learns more concise reasoning traces. We hypothesize that this happens because, unlike LiveCodeBench v6, these domains are not reasoning intensive, and thus do not require the small-KL-shift property RL offers (Section 3.1), which is needed for strong generalization in reasoning problems (Shenfeld et al., 2025; Chu et al., 2025). To test this, we compare SDPO against a baseline that does no reasoning at all: SFT on *final answer strings with no chain-of-thought*. If this baseline matches SDPO, SDPO's gains over GRPO *need not reflect improved reasoning*.

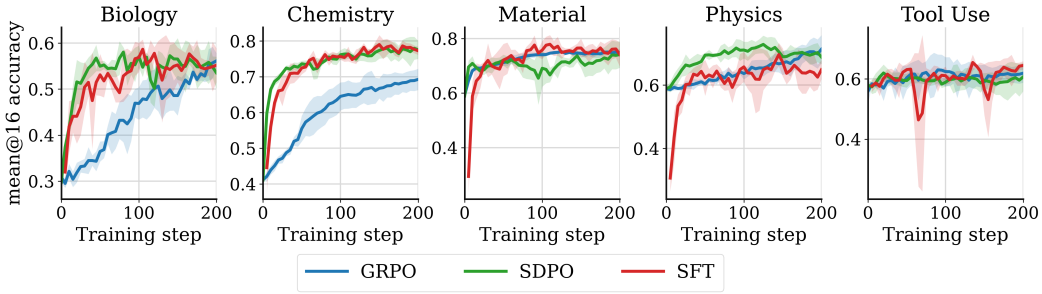


Figure 4: GRPO, SDPO, and SFT Qwen3-8B validation performance on SciKnowEval and ToolAlpaca datasets; mean±std accuracy from four seeds is plotted.

Setup. Hübotter et al. (2026) evaluate SDPO on SciKnowEval multiple-choice science questions (Feng et al., 2024) and ToolAlpaca (Tang et al., 2023), with teachers conditioned on correct neighboring solutions. Unlike math, where reaching the answer requires multi-step derivation, answers in these domains can be directly produced by choosing among four options or emitting an API call whose structure is largely determined by the prompt. Concretely, we train Qwen3-8B via SFT for 200 steps to respond directly with `<answer>A/B/C/D</answer>` on SciKnowEval and with the ground-truth API call on ToolAlpaca. We compare this against SDPO and GRPO evaluating on a held-out validation set; we report our hyperparameters in Appendix E.1.

Final-answer SFT matches or exceeds SDPO performance in 4/5 domains. Figure 4 and Table 7 in Appendix D.2 show that final-answer SFT matches or exceeds SDPO in every domain except physics, where SDPO reaches the highest peak accuracy but GRPO matches its final performance at 200 steps. Since chain-of-thought is not needed to achieve high validation accuracy, these domains do not appear to be reasoning intensive, consistent with our hypothesis.

Final-answer SFT beats SDPO in chemistry, the only domain where SDPO clearly beats GRPO. In the other four domains, SDPO stays within 1.9 points of GRPO, leading in biology and physics and trailing in materials and tool use. In chemistry, SDPO improves on GRPO by 8.7 points (78.0 vs. 69.3), but final-answer SFT also improves by 9.7 (79.0). SDPO's one clear gain thus coincides with the largest gain of a model that does no reasoning. Physics is the only domain where final-answer SFT falls below both GRPO and SDPO performance, although these gaps are comparable to the standard deviations we record.

SDPO and SFT peak early, while GRPO improves more slowly. The training curves in Figure 4 show a second similarity between SDPO and final-answer SFT: both reach their peak validation accuracy early in training while GRPO improves more gradually over the 200 steps. This is consistent with SDPO directly fitting the answer rather than improving gradually via reward-driven learning, and mirrors its faster early progress on the training problems of LiveCodeBench v6 (Section 4.1).

SDPO's strengths may therefore lie in *reasoning-free* settings, in line with its reported gains in human preference alignment (Buening et al., 2026).

4.3 OPSD GAINS DEPEND ON THINKING MODE TRANSFER, NOT PRIVILEGED INFORMATION

OPSD (Zhao et al., 2026a) reports gains on AIME24, AIME25, and HMMT25 (Mathematical Association of America, 2024; Balunović et al., 2025) by distilling a thinking-on Qwen3 teacher conditioned on a reference solution z into the same model with thinking off (Appendix C). The teacher thus differs from the student in two ways: it sees z, and it thinks. If the gains come from conditioning on z, replacing it with the reference solution of a different problem or removing it entirely should degrade performance.

Setup. Following Zhao et al. (2026a), we train on OpenThoughts (Guha et al., 2025) math problems, each paired with a reference solution. We fix the student to thinking-off and ablate the teacher's thinking mode (on or off) and its privileged information: the problem's own reference solution (ref z), no reference solution (no z), or the reference solution of a different problem (shuffled z). The teacher template is otherwise fixed, so the three differ only in the content of z.

Zhao et al. (2026a) train Qwen3-1.7B to 8B with OPSD for 100 steps and compare against thinking-on GRPO (500 steps) and SFT (100 steps). To evaluate fully converged performance, we train OPSD and SFT for 200 steps and GRPO, with thinking on and off, for 500 steps, and evaluate on the same datasets with thinking on and off. As Shenfeld et al. (2026); Hübotter et al. (2026) suggest self-distillation works better for larger models, we also test Qwen3-14B and a second model family, Olmo-3-7B-Instruct. As the latter has no thinking toggle, we instead compare two teachers given the same privileged information: the student itself (self-distillation) and Olmo-3-7B-Think, a separately trained thinking model from the same family (cross-model distillation), which cleanly separates privileged information from thinking-mode transfer. We report Qwen3-8B validation results in Table 1, remaining results and statistics in Appendix D.3, and hyperparameters in Table 24.

Table 1: Qwen3-8B training methods evaluated under `avg@12` with thinking-on and thinking-off evaluation. OPSD think/no think denote whether teacher thinking was on during OPSD training. The subscript on GRPO runs denotes whether thinking was on or off during training. Denote z as the reference solution shown to the teacher. For every evaluation mode, we report the checkpoint with the best average performance (averaged over four seeds), reporting mean±std accuracy.

	Thinking-on eval				**Thinking-off eval**			
Method	**AIME24**	**AIME25**	**HMMT25**	**Avg.**	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Baselines								
Base	75.6	68.6	45.0	63.1	26.9	21.4	9.4	19.3
$\text{GRPO}_{\text{think}}$	76.5±0.3	69.4±1.0	45.1±2.7	63.7	28.5±0.8	22.4±1.7	11.8±1.5	20.9
$\text{GRPO}_{\text{nothink}}$	76.5±0.5	70.8±1.4	45.6±1.8	64.3	34.6±2.0	23.6±1.3	12.9±1.1	23.7
SFT	76.0±1.7	70.0±0.5	43.0±1.7	63.0	**65.1±1.0**	**51.2±2.0**	**33.8±1.0**	**50.0**
OPSD (teacher thinking-on)								
Ref z	77.2±1.1	71.6±1.1	46.4±0.4	65.0	45.1±1.6	34.5±0.7	19.8±2.4	33.1
No z	78.1±1.0	71.0±1.4	46.7±1.3	**65.3**	57.6±1.2	45.6±2.0	30.6±2.9	44.6
Shuffled z	**78.3±0.6**	71.2±0.9	45.9±1.0	65.1	41.5±7.3	36.7±4.5	21.2±3.4	33.2
OPSD (teacher thinking-off)								
Ref z	77.2±0.8	70.2±1.7	46.2±1.3	64.6	26.8±1.6	22.7±0.8	9.1±0.5	19.5
No z	76.4±1.4	**72.1±1.3**	44.9±0.3	64.5	26.5±2.3	22.6±0.7	12.1±0.4	20.4
Shuffled z	77.6±0.6	70.2±0.8	**46.8±0.9**	64.9	26.9±1.9	19.2±1.5	11.2±1.7	19.1

The content of z does not matter. Holding the teacher's thinking mode and the evaluation mode fixed, we ablate z; on a sample rollout, removing it barely changes the teacher's token probabilities (Figure 2). "No z" often matches or outperforms "ref z", especially in Zhao et al. (2026a)'s orig-

inal setting (thinking teachers and thinking-on evaluations), and "ref z" achieves the best average accuracy in only three of 18 configurations, each by under one point. In the remaining thinking-on evaluations, "no z" is best with a thinking-on teacher and "shuffled z" with a thinking-off teacher. Except for thinking-off evaluations with a thinking-on teacher, the average accuracies of the three choices of z lie within 2.7 points of each other in every configuration other than Qwen3-1.7B with a non-thinking teacher (up to 5.2; Table 1, Appendix D.3). With a thinking-on teacher, thinking-off accuracy varies more across z (up to 11.7 points), but "ref z" is never the best choice: "no z" or "shuffled z" beats it by up to 11.5 points on Qwen3-8B. *At every scale, a variant lacking either the presence or the relevance of a correct reference matches "ref z", so neither is required for OPSD's gains.*

Thinking-on gains are small and do not discriminate between hypotheses. OPSD's headline results are thinking-on evaluations of students trained with thinking teachers, which Zhao et al. (2026a) attribute to teacher reasoning transfer. Beyond the smallest model, gains are modest (7.4, 0.8, 2.2, and 1.1 points for Qwen3-1.7B to Qwen3-14B for the best z), and every variant tested recovers them: thinking-off distillation (6.1, 0.8, 1.8, and 0.8), any choice of z, and, on Qwen3-8B, a loss restricted to the last 10% of tokens (Appendix D.5). GRPO trained thinking-off is also always slightly larger than thinking-on GRPO. Thinking-on gains are thus not evidence for any particular transfer mechanism, so we focus on thinking-off evaluations, the student's native inference mode.

Thinking-on teachers induce student thinking. Rather than improving student direct-answer capabilities, OPSD with a thinking teacher relies on thinking-mode transfer. Enabling teacher thinking improves average thinking-off accuracy by 7.6, 22.9, 17.3, and 26.7 points for Qwen3-1.7B to Qwen3-14B, whereas replacing ref z with shuffled z changes it by 2.15, -0.2, -0.15, and -2.6 points. Likewise, Olmo-3-7B-Instruct gains 3.5 points with a self-teacher but 13.9 with Olmo-3-7B-Think, averaged over choices of z (Table 11). Since Zhao et al. (2026a) report only results with a thinking teacher, their remaining thinking-off gains are driven by the teacher's thinking mode, not by privileged information. This transfer is visible in the students' outputs: in thinking-off evaluations, Qwen3 students distilled from a thinking teacher often emit an *orphan* `</think>` token, closing a thinking block that the chat template has already closed (Appendix C). These rollouts reason in the first person, box an answer, emit `</think>`, then write a clean solution, mirroring a thinking-on response (Figures 7 to 9). Among on-policy methods, orphan tokens occur mainly in thinking-off rollouts from OPSD with thinking teachers (up to 69.9%); GRPO and the base model produce none, and OPSD with a thinking-off teacher at most 6.7%. Thinking-on teachers also increase response length by 2.72 to 5.82 times, approaching the base model's thinking-on lengths (Table 14, Appendix D.3), whereas length stays more stable with teacher thinking disabled (0.8 to 2.52 times the base model's). Thus, with teacher thinking enabled, OPSD's thinking-off gains come from forcing thinking-mode behavior into thinking-off mode, which requires no privileged information.

4.4 Self-distillation gradients do not align with policy gradients

If the self-conditioned teacher approximated π^* (Section 3.1), as Shenfeld et al. (2026) assume, self-distillation gradients would align with the policy gradient; we test this directly.

Setup. Practical self-distillation losses differ from the idealized objectives of Proposition 3.1: they use other divergences such as Jensen–Shannon, per-token clipping, top-k logit approximations, and EMA teachers (Equation (3)), so rather than testing the proposition's equality directly, we test a consequence: a method whose teacher approximates the optimal teacher should update the model in the same direction as the policy gradient. We therefore measure the cosine similarity between each method's training gradient and a *reference*, the GRPO gradient on held-out problems, which is the policy gradient direction that increases reward on new problems. To a first order approximation, a small step along a method's gradient increases held-out reward if its cosine similarity with the reference is positive, and decreases it if negative. All gradients are estimated at the base model: on-policy RL favors high-reward policies that stay close to the base model in KL (Shenfeld et al., 2025), so we test whether the teacher is KL-optimal relative to the base model. We use the same setup as Section 4.1 and Section 4.3: for math, methods train on OpenThoughts and the reference is $\text{GRPO}_{\text{nothink}}$ on AIME24, AIME25, and HMMT25 (held-out math policy gradient); for LiveCodeBench v6, methods train on TRAIN_SEEN and the reference is GRPO on TEST_UNSEEN. We report our complete results in Appendix D.6. Since cosine similarity is a nonlinear function of

the pooled gradients and we estimate gradients over four seeds, we estimate its error with a leave-one-seed-out jackknife; we detail our gradient estimation method in Appendix E.3.

OPSD points away from the held-out math policy gradient. Across Qwen3-1.7B to 14B, $\text{GRPO}_{\text{nothink}}$ is the only method with positive cosine similarity with the held-out policy gradient on every model ($+0.018$ to $+0.166$), and clearly so on Qwen3-8B and 14B ($+0.118 \pm 0.050$ and $+0.166 \pm 0.029$; Tables 2 and 18). In contrast, all 24 OPSD variants have negative cosine similarity (-0.205 to -0.071). On Qwen3-8B, the six variants lie between -0.172 and -0.157, within each other's error bars. Notably, OPSD with a thinking-on teacher points away from the held-out policy gradient despite its large thinking-off gains (Table 1): rather than moving the student towards its own high-reward solutions like policy gradients, it improves performance by matching the teacher's thinking style (Section 4.3). Instead, on Olmo-3-7B-Instruct, GRPO and all six OPSD variants have small positive cosine similarity ($+0.026$ to $+0.057$), but their jackknife errors (0.021 to 0.052) are comparable to the values themselves, so none are clearly resolved from zero (Table 19).

Table 2: Cosine similarity with the held-out $\text{GRPO}_{\text{nothink}}$ gradient on math for Qwen3-8B (pooled over four seeds $\pm$ jackknife error).

Method	Similarity
Baselines	
$\text{GRPO}_{\text{think}}$	-0.014 ± 0.019
$\text{GRPO}_{\text{nothink}}$	$\mathbf{+0.118 \pm 0.050}$
SFT	-0.075 ± 0.031
OPSD (teacher thinking-on)	
Ref z	-0.160 ± 0.020
No z	-0.157 ± 0.019
Shuffled z	-0.157 ± 0.019
OPSD (teacher thinking-off)	
Ref z	-0.163 ± 0.013
No z	-0.172 ± 0.018
Shuffled z	-0.163 ± 0.011

SDPO gradients are less aligned than GRPO gradients. On LiveCodeBench v6, SDPO is less aligned with the held-out policy gradient than GRPO on all three models, and points away from it on Qwen3-8B (Table 3). These results are consistent with SDPO's failure to improve TEST_UNSEEN accuracy (Section 4.1), suggesting the self-distillation objective is only weakly correlated with generalizable reward maximization.

Table 3: Cosine similarity with the held-out GRPO gradient on LiveCodeBench v6 (TEST_UNSEEN) for methods trained on TRAIN_SEEN (pooled over four seeds $\pm$ jackknife error).

Model	GRPO	SDPO
Qwen3-8B	$\mathbf{+0.121 \pm 0.011}$	-0.167 ± 0.063
Qwen3-14B	$\mathbf{+0.070 \pm 0.042}$	$+0.013 \pm 0.112$
Olmo-3-7B-Instruct	$\mathbf{+0.422 \pm 0.091}$	$+0.062 \pm 0.028$

Self-distillation gradients have low variance but are misaligned. OPSD and SDPO gradients are nearly identical across seeds: their gradient noise scale (McCandlish et al., 2018), a normalized measure of variance across seeds where 0 is noiseless (Appendix E.3), is at most 0.05, versus 1.97 to 4.11 for GRPO across all models (see Appendix D.6). Self-distillation thus delivers the low-variance signal it promises, but that signal does not point along the held-out policy gradient: the self-conditioned teacher does not approximate the optimal teacher of Proposition 3.1.

5 Limitations

Our experiments are limited to the Qwen3 and Olmo model families and the SDPO and OPSD self-distillation recipes. We do not address the more recent developments on self-distillation techniques, which include harnesses (Zhao et al., 2026b), contrastive formulations (Pan et al., 2026), or more complex aggregation of feedback from neighboring rollouts (Yu et al., 2026), which may combat the limitations of these self-distillation methods. Moreover, we focus on verifiable domains and acknowledge that self-distillation has use cases in non-verifiable settings (Buening et al., 2026).

6 Conclusion

We show that the gains attributed to privileged self-distillation either do not generalize or do not come from privileged information. SDPO's improvements on LiveCodeBench v6 are confined to problems seen during training, and its gains in other domains are largely recoverable with supervised fine-tuning. OPSD's math gains rely on thinking-mode transfer, with privileged information having little effect. Finally, self-distillation gradients do not align with the policy gradient on held-out problems, indicating that the self-conditioned teacher does not approximate the optimal teacher. Together, these results highlight the need to develop principled reinforcement learning methods that can exploit privileged information.

AI USE STATEMENT

In this work, we used generative AI tools to propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, and interpret results. We did not use generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, or assist in writing proofs. Generating synthetic datasets, translation, and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures, summarize or analyze existing literature, discover research topics or identify gaps, brainstorm, search for information, identify relevant literature, propose a title or keywords, draft parts of the paper, edit the paper for readability, and create or edit software code. We have reviewed all AI-assisted work: two authors reviewed and tested LLM-generated code for correctness, and we checked all LLM-retrieved examples. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We make our code publicly available at `https://anonymous.4open.science/r/does-sd-improve-reasoning/`. We detail our experimental setups and hyperparameters for SDPO and GRPO on LiveCodeBench v6, SciKnowEval, and ToolAlpaca in Appendix E.1; for OPSD, GRPO, and SFT on OpenThoughts in Appendix E.2; for the gradient similarity analysis in Appendix E.3; and the OPSD prompts in Appendix E.4. The proof of Proposition 3.1 is in Appendix B.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova Dassarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Thomas Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv*, abs/2204.05862, 2022. URL https://api.semanticscholar.org/CorpusID:248118878.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. *arXiv preprint arXiv:2603.12273*, 2026.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dYur3yabMj.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan

Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Marina Favaro and Jack Clark. When AI builds itself: Our progress toward recursive self-improvement, and its implications. The Anthropic Institute, Anthropic, 2026. URL https://www.anthropic.com/institute/recursive-self-improvement. Updated September 18, 2026. Accessed September 22, 2026.

Kehua Feng, Keyan Ding, Weijie Wang, Xiang Zhuang, Zeyuan Wang, Ming Qin, Yu Zhao, Jianhua Yao, Qiang Zhang, and Huajun Chen. Sciknoweval: Evaluating multi-level scientific knowledge of large language models, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

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, 2025. URL https://arxiv.org/abs/2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Sarthak Harne, Chinmay Karkar, Yash Pandya, Ahmed Awadallah, and Akshay Nambi. Privileged, but biased: How pi-conditioned teachers break self-distillation. *arXiv preprint arXiv:2608.04794*, 2026.

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

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation, January 2026. URL https://arxiv.org/abs/2601.20802v2.

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, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

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

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

Kimi Team. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *DeepRLStructPred@ICLR*, 2019. URL https://api.semanticscholar.org/CorpusID:198489118.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

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.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, et al. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/maa-invitational-competitions/, 2024.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Xuan-Phi Nguyen, Zeyu Leo Liu, Yang Li, Shrey Pandit, Yiran Zhao, Anurag Koul, and Shafiq Joty. Privileged likelihood is not automatically value: Three checks for token credit in on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.09263.

Olmo Team. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

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.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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, Jyothish Pari, and Pulkit Agrawal. RL's Razor: Why On-Policy Reinforcement Learning Forgets Less. December 2025. URL https://arxiv.org/abs/2509.04259.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-Distillation Enables Continual Learning. June 2026. URL https://arxiv.org/abs/2601.19897.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Samyak Shrestha and Alexander Tessier. Rethinking privileged information in on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.18271.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12, pp. 1057–1063. MIT Press, 2000.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases, 2023.

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

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training, 2026. URL https://arxiv.org/abs/2607.01763.

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

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.emnlp-demos.6/.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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

Weichen Yu, Xiaomin Li, Yizhou Zhao, Xiaoze Liu, Ruowang Zhang, Haixin Wang, Yinyi Luo, Chen Henry Wu, Gaurav Mittal, Matt Fredrikson, et al. Multi-rollout on-policy distillation via peer successes and failures. *arXiv preprint arXiv:2605.12652*, 2026.

XiuYu Zhang, Wei Chow, Junfeng Fang, Xingyu Zhu, Zhenkai Liang, and Tat-Seng Chua. What does privileged information add to on-policy self-distillation?, 2026. URL https://arxiv.org/abs/2609.20612.

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. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=Jpxfof0EaS.

Zhengyang Zhao, Lu Ma, and Wentao Zhang. Training with harnesses: On-policy harness self-distillation for complex reasoning, 2026b. URL https://arxiv.org/abs/2605.08741.

A Extended related works

We divide existing work on the limitations of self-distillation techniques into the following:

Empirical re-evaluations. Wang et al. (2026) demonstrate that SDPO struggles to generalize to out-of-distribution domains, and fail in continual learning settings where GRPO succeeds. Kaur et al. (2026) highlight that self-distillation degrades thinking-models; with previous works primarily focusing on instruction-tuned models. Concurrent to our work, Shrestha & Tessier (2026); Zhang et al. (2026); Ichihara et al. (2026) show that OPSD does not rely on privileged information, but on thinking-mode transfer. In contrast to these works, we provide our results with an *exact* reproduction of SDPO and OPSD training and evaluating settings, and we ask whether SDPO's gains in other domains reflect improvements in reasoning.

Token-level symptoms. Kim et al. (2026); Kaur et al. (2026); Pan et al. (2026); Harne et al. (2026) show that self-distillation losses concentrate on tokens which express uncertainty and have high entropy. This results in a suppression of the student's backtracking behavior and collapse in response lengths. Instead, we focus on the positive results reported by OPSD, and find through token-level diagnoses and response length metrics, that performance improvements relate to thinking-mode transfer.

Sequence-level diagnoses. Nguyen et al. (2026) develop statistics that indicate that the self-conditioned teacher cannot differentiate between correct and incorrect solutions generated by the student. Harne et al. (2026) present analogous results, showing that the self-conditioned teacher is greatly biased towards the reference solution it is provided, mirroring Yang et al. (2026)'s results indicating that the teacher leaks privileged information. On the other hand, we diagnose whether the self-conditioned teacher approximates the optimal KL-regularized model, which is an assumption behind prior self-distillation work (Shenfeld et al., 2026).

Remedies. Pan et al. (2026) and Yang et al. (2026) retain GRPO as a learning signal and use the self-conditioned teacher only to modulate its per-token advantages. Because these methods do not rely on the self-distillation loss alone, which is the setting we study, they fall outside the scope of this work. Their reliance on GRPO is consistent with Harne et al. (2026), who question whether the self-distillation objective on its own, without such auxiliary machinery, yields any performance gains at all.

B Proof of Proposition 3.1

We restate the proposition for convenience.

Proposition 3.1. *Consider the single-prompt setting, with a fixed $x \sim \rho(\cdot)$. Fix any policy $\pi_\theta(\cdot|x)$ over $\mathcal{Y}$, a binary reward function $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, and $\beta > 0$, and define the optimal teacher $\pi^*(y|x) = \frac{1}{Z(x)}\pi_\theta(y|x)\exp(R(x, y)/\beta)$, which maximizes $\mathbb{E}_{y\sim\pi(\cdot|x)}[R(x, y)] - \beta D_{\mathrm{KL}}(\pi(\cdot|x) \,\|\, \pi_\theta(\cdot|x))$ over policies π. Since x is fixed, we suppress it from the notation, writing $\pi_\theta(y)$, $R(y)$, Z, and $\pi^*(y) = \frac{1}{Z}\pi_\theta(y)\exp(R(y)/\beta)$. Then, we have that:*

$$\frac{\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]}{\|\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]\|_2} = -\frac{\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}\,(\pi^*) \,\|\, \pi_\theta)}{\|\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}\,(\pi^*) \,\|\, \pi_\theta)\|_2} = -\frac{\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \mathrm{sg}\,(\pi^*))}{\|\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \mathrm{sg}\,(\pi^*))\|_2}, \tag{4}$$

whenever $\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[R(y)]$ is non-zero.

Proof. We use the notation of the proposition, suppressing x. Since $\mathcal{Y}$ is finite, all expectations over $\mathcal{Y}$ are finite sums, so gradients commute with expectations, and we have the log-derivative identity $\nabla_\theta \pi_\theta(y) = \pi_\theta(y)\nabla_\theta \log \pi_\theta(y)$ and the score identity $\mathbb{E}_{y\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(y)] = \sum_y \nabla_\theta \pi_\theta(y) = 0$. Let $J(\theta) = \mathbb{E}_{y\sim\pi_\theta}[R(y)]$ and $g = \nabla_\theta J(\theta)$. We show that both KL gradients are negative multiples of g.

Policy gradient. By the log-derivative identity,

$$g = \mathbb{E}_{y\sim\pi_\theta}\left[R(y)\,\nabla_\theta \log \pi_\theta(y)\right].$$

Reverse KL. Since $\mathrm{sg}(\pi^*)$ is constant in θ,

$$\begin{aligned}
\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \mathrm{sg}(\pi^*)) &= \sum_y (\nabla_\theta \pi_\theta(y)) \log \frac{\pi_\theta(y)}{\pi^*(y)} + \sum_y \pi_\theta(y)\, \nabla_\theta \log \pi_\theta(y) \\
&= \mathbb{E}_{y\sim\pi_\theta}\left[\log \frac{\pi_\theta(y)}{\pi^*(y)}\, \nabla_\theta \log \pi_\theta(y)\right] && \text{(score identity)} \\
&= \mathbb{E}_{y\sim\pi_\theta}\left[(\log Z - R(y)/\beta)\, \nabla_\theta \log \pi_\theta(y)\right] && \text{(definition of } \pi^*) \\
&= -\frac{1}{\beta}\, g. && \text{(score identity)}
\end{aligned}$$

Forward KL. Since $\pi^*(y) > 0$ if and only if $\pi_\theta(y) > 0$, we may change measure from π^* to π_θ:

$$\begin{aligned}
\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}(\pi^*) \,\|\, \pi_\theta) &= -\mathbb{E}_{y\sim\pi^*}\left[\nabla_\theta \log \pi_\theta(y)\right] \\
&= -\mathbb{E}_{y\sim\pi_\theta}\left[\frac{\pi^*(y)}{\pi_\theta(y)}\, \nabla_\theta \log \pi_\theta(y)\right] \\
&= -\frac{1}{Z}\, \mathbb{E}_{y\sim\pi_\theta}\left[\exp(R(y)/\beta)\, \nabla_\theta \log \pi_\theta(y)\right].
\end{aligned}$$

Because $R(y) \in \{0, 1\}$, we have $\exp(R(y)/\beta) = 1 + (e^{1/\beta} - 1)R(y)$. Substituting and applying the score identity to the constant term,

$$\nabla_\theta D_{\mathrm{KL}}(\mathrm{sg}(\pi^*) \,\|\, \pi_\theta) = -\frac{e^{1/\beta} - 1}{Z}\, g.$$

Conclusion. Both $1/\beta$ and $(e^{1/\beta} - 1)/Z$ are strictly positive for $\beta > 0$. Hence, whenever $g \neq \vec{0}$, the negated gradients of both KL objectives are positive multiples of g, and normalizing each to unit ℓ_2 norm yields Equation (4). □

C QWEN3 THINKING MODES

Qwen3 models support a thinking and a non-thinking mode in the *same model* (Qwen Team, 2025). For the flagship models (Qwen3-32B and Qwen3-235B-A22B), the two modes are merged during post-training (*Thinking Mode Fusion*) by fine-tuning on a mixture of "thinking" and "non-thinking" data. The smaller models (Qwen3 0.6B to 14B and Qwen3-30B-A3B) inherit both of these modes through supervised fine-tuning and on-policy-distillation from the flagship models. Concretely, in thinking mode, the model reasons inside a `<think>...</think>` block before responding. The mode is toggled via the `enable_thinking` argument of the chat template. When it is set to False, an empty think block is prefilled in the assistant turn:

```
<|im_start|>assistant
<think>

</think>
```

In non-thinking mode, the model is trained to respond directly after this empty thinking block.

D EXTENDED EXPERIMENTAL RESULTS

D.1 SDPO LIVECODEBENCH V6 PERFORMANCE

Figures 5 and 6 show the LiveCodeBench v6 training curves for Qwen3-14B and Olmo-3-7B-Instruct, complementing Figure 3 for Qwen3-8B. Tables 4 to 6 also reports the scores of all three models on TEST_UNSEEN, TRAIN_SEEN, and TRAIN_UNSEEN at training steps 0, 80, and 200.

D.2 SDPO SCIENCE REASONING AND TOOL USE PERFORMANCE

Table 7 reports the best-checkpoint validation accuracy of GRPO, SDPO, and final-answer SFT on SciKnowEval and ToolAlpaca, corresponding to the training curves in Figure 4 (Section 4.2).

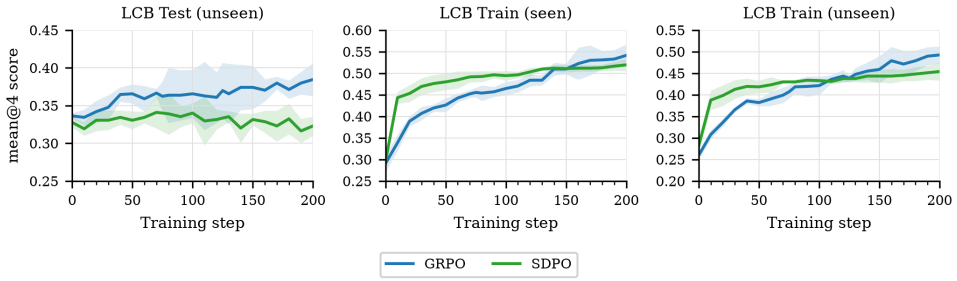


Figure 5: Qwen3-14B SDPO vs. GRPO on LiveCodeBench v6 across four seeds, scored as the fraction of problems with no failing test cases (mean ± std). TRAIN_SEEN and TRAIN_UNSEEN evaluate problems seen during training, on seen and on held-out test cases respectively; TEST_UNSEEN evaluates held-out problems.



Figure 6: Olmo-3-7B-Instruct SDPO vs. GRPO on LiveCodeBench v6 across four seeds, scored as the fraction of problems with no failing test cases (mean ± std). TRAIN_SEEN and TRAIN_UNSEEN evaluate problems seen during training, on seen and on held-out test cases respectively; TEST_UNSEEN evaluates held-out problems.

Table 4: LiveCodeBench v6 score (fraction of problems with no failing test cases, mean±std over four seeds) on TEST_UNSEEN for SDPO and GRPO at initialization, 80 steps, and 200 steps. Bold marks the better method at each step.

Model	Method	Step 0	Step 80	Step 200
Qwen3-8B	SDPO	**32.3±1.6**	30.1±0.7	29.8±1.1
	GRPO	30.6±1.0	**33.3±1.9**	**34.3±0.5**
Qwen3-14B	SDPO	32.8±0.7	33.9±2.8	32.3±1.3
	GRPO	**33.6±0.4**	**36.4±3.7**	**38.5±2.2**
Olmo-3-7B-Instruct	SDPO	30.9±1.0	25.2±1.0	21.0±1.4
	GRPO	**31.5±1.8**	**32.7±1.2**	**34.1±0.8**

Table 5: LiveCodeBench v6 score (fraction of problems with no failing test cases, mean±std over four seeds) on TRAIN_SEEN for SDPO and GRPO at initialization, 80 steps, and 200 steps. Bold marks the better method at each step.

Model	Method	Step 0	Step 80	Step 200
Qwen3-8B	SDPO	**25.8±1.8**	**48.6±0.7**	48.9±1.6
	GRPO	25.1±0.7	43.8±2.4	**50.9±1.1**
Qwen3-14B	SDPO	**29.4±0.4**	**49.2±1.3**	51.9±1.5
	GRPO	29.0±2.0	45.4±1.8	**54.1±2.5**
Olmo-3-7B-Instruct	SDPO	**28.9±1.7**	**52.9±2.3**	**47.2±4.7**
	GRPO	28.0±1.9	39.7±1.3	43.7±2.4

Table 6: LiveCodeBench v6 score (fraction of problems with no failing test cases, mean±std over four seeds) on TRAIN_UNSEEN for SDPO and GRPO at initialization, 80 steps, and 200 steps. Bold marks the better method at each step.

Model	Method	Step 0	Step 80	Step 200
Qwen3-8B	SDPO	**25.6±0.5**	**44.1±1.0**	44.0±1.7
	GRPO	25.4±1.0	41.2±2.4	**46.2±2.0**
Qwen3-14B	SDPO	**28.2±1.4**	**43.0±0.2**	45.4±2.1
	GRPO	26.0±1.0	41.8±1.6	**49.2±2.0**
Olmo-3-7B-Instruct	SDPO	**25.9±1.6**	**48.5±1.4**	**42.6±6.2**
	GRPO	25.8±0.4	36.2±2.3	41.9±0.8

Table 7: GRPO, SDPO, and SFT Qwen3-8B validation performance on SciKnowEval and ToolAlpaca datasets. The accuracy of the best checkpoint, evaluated by mean accuracy across four seeds, is reported.

Method	Biology	Chemistry	Material	Physics	Tool use
GRPO	56.1±1.7	69.3±3.0	75.2±0.3	71.4±4.4	63.0±5.1
SDPO	58.0±1.1	78.0±2.4	74.1±4.8	**72.9±2.8**	62.0±2.8
SFT	**58.7±2.1**	**79.0±2.7**	**77.9±3.4**	69.6±3.7	**65.4±2.6**

D.3 OPSD METRICS ON QWEN3 MODELS

Table 8: Qwen3-1.7B training methods evaluated under `avg@12` with thinking-on and thinking-off evaluation. OPSD think/no think denote whether teacher thinking was on during OPSD training. The subscript on GRPO runs denotes whether thinking was on or off during training. Denote z as the reference solution shown to the teacher. For every evaluation mode, we report the checkpoint with the best average performance (averaged over four seeds), reporting mean±std accuracy.

	Thinking-on eval				**Thinking-off eval**			
Method	**AIME24**	**AIME25**	**HMMT25**	**Avg.**	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Baselines								
Base	46.7	34.4	21.9	34.4	15.8	10.0	4.4	10.1
$\text{GRPO}_{\text{think}}$	49.9±3.5	36.3±1.3	23.5±3.1	36.6	15.3±1.8	8.7±1.9	5.9±1.4	10.0
$\text{GRPO}_{\text{nothink}}$	49.9±2.1	37.8±1.7	24.7±1.1	37.4	14.7±1.1	11.4±0.6	6.7±0.6	10.9
SFT	50.3±0.8	36.9±1.3	25.8±1.4	37.6	**37.8±3.3**	**27.6±1.2**	**18.3±2.1**	**27.9**
OPSD (teacher thinking-on)								
Ref z	**56.0±1.2**	**42.4±1.6**	27.1±0.5	**41.8**	26.7±2.0	19.7±1.7	12.2±1.3	19.5
No z	53.0±2.3	39.7±1.0	26.5±0.9	39.7	16.4±1.4	13.1±0.7	7.4±0.9	12.3
Shuffled z	**56.0±1.4**	39.2±0.7	**27.8±0.5**	41.0	28.8±1.6	21.5±1.0	16.0±1.2	22.1
OPSD (teacher thinking-off)								
Ref z	54.1±1.9	39.7±0.9	27.2±1.2	40.3	15.2±0.7	11.5±1.5	6.4±0.7	11.0
No z	49.0±2.8	36.9±0.9	21.8±1.3	35.9	11.2±0.8	7.2±0.8	4.2±0.7	7.5
Shuffled z	54.0±0.8	40.1±2.0	27.4±1.7	40.5	16.4±1.6	13.2±1.4	8.5±0.3	12.7

Table 9: Qwen3-4B training methods evaluated under `avg@12` with thinking-on and thinking-off evaluation. OPSD think/no think denote whether teacher thinking was on during OPSD training. The subscript on GRPO runs denotes whether thinking was on or off during training. Denote z as the reference solution shown to the teacher. For every evaluation mode, we report the checkpoint with the best average performance (averaged over four seeds), reporting mean±std accuracy.

	Thinking-on eval				**Thinking-off eval**			
Method	**AIME24**	**AIME25**	**HMMT25**	**Avg.**	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Baselines								
Base	**76.4**	67.8	43.1	62.4	21.7	20.3	11.9	18.0
$\text{GRPO}_{\text{think}}$	73.3±0.9	67.0±1.7	44.3±1.6	61.6	23.6±0.9	20.8±1.0	11.7±0.9	18.7
$\text{GRPO}_{\text{nothink}}$	74.3±0.3	66.2±0.8	44.9±0.7	61.8	28.3±1.6	22.6±0.8	14.4±0.6	21.8
SFT	74.1±1.6	68.3±1.1	42.3±1.1	61.6	**61.9±2.5**	50.7±1.4	**32.9±1.5**	**48.5**
OPSD (teacher thinking-on)								
Ref z	76.1±1.0	67.2±0.9	44.9±1.1	62.7	54.4±2.1	**51.9±2.2**	31.1±2.0	45.8
No z	74.5±2.0	**70.0±2.2**	45.0±0.8	**63.2**	42.0±0.7	37.3±2.0	24.0±1.1	34.4
Shuffled z	75.6±1.9	67.2±0.6	**46.1±1.4**	62.9	55.9±2.6	51.5±3.5	31.0±0.8	46.1
OPSD (teacher thinking-off)								
Ref z	76.2±0.5	66.9±1.3	43.6±1.5	62.2	27.3±1.7	21.6±0.8	12.1±1.0	20.3
No z	74.0±0.3	66.5±1.8	43.1±1.1	61.2	22.6±0.9	19.3±1.7	11.3±0.1	17.8
Shuffled z	76.2±1.1	68.1±2.2	45.2±0.7	**63.2**	26.2±1.8	22.6±1.6	10.1±0.7	19.6

Table 10: Qwen3-14B training methods evaluated under `avg@12` with thinking-on and thinking-off evaluation. OPSD think/no think denote whether teacher thinking was on during OPSD training. The subscript on GRPO runs denotes whether thinking was on or off during training. Denote z as the reference solution shown to the teacher. For every evaluation mode, we report the checkpoint with the best average performance (averaged over four seeds), reporting mean±std accuracy.

	Thinking-on eval				**Thinking-off eval**			
Method	**AIME24**	**AIME25**	**HMMT25**	**Avg.**	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Baselines								
Base	80.0	73.9	49.2	67.7	29.7	23.3	12.2	21.8
$\text{GRPO}_{\text{think}}$	80.8±1.1	**74.0±0.6**	50.1±1.0	68.3	31.1±1.1	24.6±0.5	12.6±0.6	22.8
$\text{GRPO}_{\text{nothink}}$	80.7±1.7	73.5±0.8	51.2±1.8	68.5	30.7±1.8	24.7±2.4	13.6±0.5	23.0
SFT	79.7±1.0	71.6±1.1	49.6±0.9	66.9	65.7±1.2	54.7±1.7	33.8±1.4	51.4
OPSD (teacher thinking-on)								
Ref z	80.8±0.6	72.7±1.3	49.7±1.5	67.7	65.3±3.8	51.8±6.5	35.4±2.4	50.8
No z	**82.3±1.1**	73.1±1.4	51.0±2.3	**68.8**	**66.3±3.3**	**59.9±1.8**	**41.0±2.9**	**55.7**
Shuffled z	80.4±0.9	72.6±0.5	50.6±2.4	67.9	59.2±1.9	47.5±1.8	32.4±1.3	46.3
OPSD (teacher thinking-off)								
Ref z	80.7±1.0	72.6±1.1	50.0±2.5	67.8	32.6±1.4	25.6±0.8	13.1±1.1	23.8
No z	81.5±1.1	73.9±0.8	50.0±1.3	68.4	34.2±1.9	26.8±0.7	16.5±1.4	25.8
Shuffled z	80.5±1.4	73.3±1.1	**51.8±1.1**	68.5	32.4±2.0	24.7±1.5	12.1±1.6	23.1

Table 11: Olmo-3-7B-Instruct training methods evaluated under `avg@12`. OPSD self-distill uses the frozen Olmo-3-7B-Instruct as the teacher, while OPSD cross-model uses Olmo-3-7B-Think as the teacher. Denote z as the reference solution shown to the teacher. We report the checkpoint with the best average performance (averaged over four seeds), reporting mean±std accuracy.

Method	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Baselines				
Base	50.3	37.2	23.6	37.0
GRPO	51.9±1.3	39.2±1.0	25.6±2.3	38.9
SFT	53.1±0.5	40.4±0.9	24.7±1.2	39.4
OPSD (self-distill)				
Ref z	55.4±1.0	41.7±0.4	26.4±0.8	41.2
No z	54.0±1.7	41.7±1.3	25.3±0.9	40.3
Shuffled z	53.3±2.3	41.4±1.4	25.6±1.6	40.1
OPSD (cross-model)				
Ref z	64.6±2.5	50.7±1.1	35.9±2.2	50.4
No z	**66.1±1.3**	**52.3±2.1**	**38.1±1.3**	**52.2**
Shuffled z	64.6±0.4	51.0±1.1	35.0±2.4	50.2

Table 12: Qwen3-1.7B average response length and percentage of rollouts with an orphan </think> tokens generated per method. Statistics are pooled over the rollouts generated from the existing AIME 24, AIME 25, and HMMT 25 evaluations at the best checkpoint when performance was averaged over four seeds. We report mean±std over seeds. Tok@best refers to the average response length in tokens of the best checkpoint, Tok@200 is the same quantity at checkpoint 200, and Orph.% refers to the fraction of thinking-off rollouts generated by the best checkpoint that emit a </think> with no opener of their own (we do not compute this for thinking-on as the model emits its own <think> and the closer is always matched).

	Thinking-on eval		**Thinking-off eval**		
Method	**Tok@200**	**Tok@best**	**Orph. %**	**Tok@200**	**Tok@best**
Baselines					
Base	18413	18413	0.0	2889	2889
$\text{GRPO}_{\text{think}}$	18220±1459	18356±743	0.0±0.0	2985±374	2827±145
$\text{GRPO}_{\text{nothink}}$	19089±1475	19517±1849	0.0±0.0	3161±874	3187±580
SFT	14646±219	20023±160	48.4±1.5	15438±118	15438±118
OPSD (teacher thinking-on)					
Ref z	20526±224	21301±271	31.7±1.0	10602±157	10602±157
No z	20991±356	20967±111	3.2±0.8	9887±978	9534±773
Shuffled z	19821±288	19933±102	42.2±1.0	11397±422	11456±243
OPSD (teacher thinking-off)					
Ref z	18869±217	20026±166	0.3±0.2	3442±311	4008±104
No z	18553±295	17374±165	0.0±0.0	4816±3008	2381±197
Shuffled z	17619±45	21258±66	0.7±0.2	3121±208	5470±231

Table 13: Qwen3-4B average response length and orphan </think> tokens generated per method. Statistics are pooled over the rollouts generated from the existing AIME 24, AIME 25, and HMMT 25 evaluations at the best checkpoint when performance was averaged over four seeds. We report mean±std over seeds. Tok@best refers to the average response length in tokens of the best checkpoint and Tok@200 is the same quantity at checkpoint 200. Orph.% refers to the fraction of rollouts generated by the best checkpoint with thinking disabled that emit a </think> with no opener of their own (we do not compute this for thinking-on as the model emits its own <think> and the closer is always matched).

	Thinking-on eval		**Thinking-off eval**		
Method	**Tok@200**	**Tok@best**	**Orph. %**	**Tok@200**	**Tok@best**
Baselines					
Base	17556	17556	0.0	3719	3719
$\text{GRPO}_{\text{think}}$	17742±197	17456±106	0.0±0.0	3604±170	3746±277
$\text{GRPO}_{\text{nothink}}$	17855±350	18078±585	0.0±0.0	4348±652	4693±537
SFT	13871±153	18273±220	53.0±1.7	14028±70	14028±70
OPSD (teacher thinking-on)					
Ref z	19814±161	19832±129	64.1±1.6	16518±339	16445±308
No z	20711±205	20310±193	20.3±3.2	18560±759	15733±502
Shuffled z	19542±302	19789±51	69.9±2.6	16758±322	16732±126
OPSD (teacher thinking-off)					
Ref z	19666±161	19666±161	1.9±0.3	7138±209	7127±372
No z	18161±207	17082±114	0.0±0.0	4621±177	3395±126
Shuffled z	19368±145	19879±118	2.0±0.2	6670±135	6670±135

Table 14: Qwen3-8B average response length and orphan </think> tokens generated per method. Statistics are pooled over the rollouts generated from the existing AIME 24, AIME 25, and HMMT 25 evaluations at the best checkpoint when performance was averaged over four seeds. We report mean±std over seeds. Tok@best refers to the average response length in tokens of the best checkpoint and Tok@200 is the same quantity at checkpoint 200. Orph.% refers to the fraction of rollouts generated by the best checkpoint with thinking disabled that emit a </think> with no opener of their own (we do not compute this for thinking-on as the model emits its own <think> and the closer is always matched).

	Thinking-on eval		**Thinking-off eval**		
Method	**Tok@200**	**Tok@best**	**Orph. %**	**Tok@200**	**Tok@best**
Baselines					
Base	18264	18264	0.0	3580	3580
$\text{GRPO}_{\text{think}}$	18209±484	18160±417	0.0±0.0	3762±123	3762±123
$\text{GRPO}_{\text{nothink}}$	18453±203	18479±243	0.0±0.0	4116±367	4355±313
SFT	17807±86	18511±254	75.0±1.0	14488±143	14456±206
OPSD (teacher thinking-on)					
Ref z	20255±167	20188±131	30.6±2.5	12025±223	9750±384
No z	20174±123	20284±189	40.1±3.5	15493±447	15493±447
Shuffled z	20008±131	19669±524	25.2±6.3	13864±636	13864±636
OPSD (teacher thinking-off)					
Ref z	19460±151	19403±45	4.7±1.3	5870±110	5384±241
No z	19003±146	19003±146	0.0±0.0	6736±544	5944±230
Shuffled z	19385±157	19356±156	6.7±0.3	5710±252	9027±160

Table 15: Qwen3-14B average response length and orphan </think> tokens generated per method. Statistics are pooled over the rollouts generated from the existing AIME 24, AIME 25, and HMMT 25 evaluations at the best checkpoint when performance was averaged over four seeds. We report mean±std over seeds. Tok@best refers to the average response length in tokens of the best checkpoint and Tok@200 is the same quantity at checkpoint 200. Orph.% refers to the fraction of rollouts generated by the best checkpoint with thinking disabled that emit a </think> with no opener of their own (we do not compute this for thinking-on as the model emits its own <think> and the closer is always matched).

	Thinking-on eval		**Thinking-off eval**		
Method	**Tok@200**	**Tok@best**	**Orph. %**	**Tok@200**	**Tok@best**
Baselines					
Base	16757	16757	0.0	3341	3341
$\text{GRPO}_{\text{think}}$	16435±223	16447±144	0.0±0.0	3389±142	3438±79
$\text{GRPO}_{\text{nothink}}$	16638±200	16646±270	0.0±0.0	3500±357	3569±89
SFT	17061±138	17597±160	48.4±1.6	13145±117	13119±74
OPSD (teacher thinking-on)					
Ref z	17484±109	17525±128	55.7±10.4	13609±1780	13609±1780
No z	18370±23	18179±202	67.7±6.2	20468±532	19456±631
Shuffled z	17529±48	17568±42	44.9±5.6	15038±1884	15038±1884
OPSD (teacher thinking-off)					
Ref z	17449±125	17576±194	0.0±0.0	8608±411	5998±273
No z	17460±139	17636±69	0.0±0.0	8522±1134	6526±791
Shuffled z	17365±102	17360±92	0.0±0.0	8759±326	6156±107

Table 16: Olmo-3-7B-Instruct average response length and orphan </think> tokens generated per method. Statistics are pooled over the rollouts generated from the existing AIME 24, AIME 25, and HMMT 25 evaluations at the best checkpoint when performance was averaged over four seeds. We report mean±std over seeds. Tok@best refers to the average response length in tokens of the best checkpoint and Tok@200 is the same quantity at checkpoint 200. Orph.% refers to the fraction of rollouts generated by the best checkpoint with thinking disabled that emit a </think> with no opener of their own. Olmo-3-7B-Instruct's chat template already ignores enable_thinking (every variant renders an identical prompt with no tag prefilled, hence no thinking-on column), and its native format is to reason with no opener and then emit </think> before a clean final answer, so its base model already scores 63.0% and emits <think> in 0.0% of rollouts.

	Thinking-off eval		
Method	**Orph. %**	**Tok@200**	**Tok@best**
Baselines			
Base	63.0	7732	7732
GRPO	63.7±1.7	7888±134	7915±214
SFT	69.3±1.3	8794±175	8646±71
OPSD (self teacher)			
Ref z	56.4±1.3	7501±131	8262±101
No z	56.2±1.0	7266±110	8427±78
Shuffled z	60.4±0.5	7505±139	8015±67
OPSD (teacher Olmo-3-7B-Think)			
Ref z	99.1±0.3	14239±84	14239±84
No z	98.4±0.4	14745±24	14745±24
Shuffled z	98.8±0.2	14280±164	14280±164

D.4 OPSD EXAMPLE GENERATIONS

Problem: There exist real numbers x and y, both greater than 1, such that $\log_x(y^x) = \log_y(x^{4y}) = 10$. Find xy.

Model output (thinking-off):

```
We are given the following equations involving logarithms:

$$
\log_x(y^x) = 10 \quad \text{and} \quad \log_y(x^{4y}) = 10
$$

We are to find the value of $ xy $, where $ x > 1 $ and $ y > 1 $ are
real numbers.

[...  solution continues ...]

### Step 3:  Multiply equations (3) and (4)

Let's multiply both equations together:

$$
\left(x \cdot \frac{\log y}{\log x} \right) \cdot \left(4y \cdot
\frac{\log x}{\log y} \right) = 10 \cdot 10 = 100
$$

Simplify the left-hand side:

$$
x \cdot 4y \cdot \frac{\log y}{\log x} \cdot \frac{\log x}{\log y} = 4xy
$$

So:

$$
4xy = 100 \Rightarrow xy = \frac{100}{4} = 25
$$

---

### Final Answer:

$$
\boxed{25}
$$
```

Figure 7: Base Qwen3-8B thinking-off rollout on an AIME24 problem generated with temperature 1.0. The chat template opens a <think> block, the model reasons in the first person, tests several numerical values of x before finding a one-line argument, and closes the block with a single matched </think> before writing a polished solution. Exactly one <think> and one </think> appear in the generation. The full generation is 559 tokens long.

Problem: There exist real numbers x and y, both greater than 1, such that $\log_x\left(y^x\right) = \log_y\left(x^{4y}\right) = 10$. Find xy.

Model output (thinking-on):

```
<think>
Okay, so I need to solve this problem where there are real numbers
x and y, both greater than 1, such that log base x of y^x equals
log base y of x^{4y} equals 10.  And I need to find the product xy.
Alright, let me try to unpack this step by step.

First, let me recall some logarithm properties.  The logarithm log_b(a)
is equal to ln(a)/ln(b) if I remember correctly.  Also, there's a
power rule:  log_b(a^c) = c*log_b(a).  [...  reasoning continues ...]

What if x is smaller?  Let's try x = 2.  Let me check again.  Wait,
tried x = 2, got around 7.28.  Let me try x = 1.5.  Then y = 25 / 1.5
≈ 16.666...

Compute log_{1.5}(y^1.5) = 1.5 * log_{1.5}(y).  log_{1.5}(y) =
log_{1.5}(25 / 1.5) = log_{1.5}(25) - log_{1.5}(1.5) = log_{1.5}(25) - 1.
Let me compute log_{1.5}(25).  Let me use natural logs:  ln(25)/ln(1.5)
≈ 3.2189 / 0.4055 ≈ 7.936.  So log_{1.5}(y) ≈ 7.936 - 1 = 6.936.
Multiply by 1.5:  ≈ 10.404.  That's close to 10.  Maybe with x
slightly larger than 1.5?

Wait, so with x = 1.5, the value is approximately 10.4, which is
slightly above 10.  Let me try x = 1.6.  [...  solution continues ...]

Therefore, I think that's solid.  I can't find any mistake in this
reasoning.  So I think the answer is 25.

**Final Answer**
\boxed{25}
</think>

To solve the given problem, we are given two equations involving
logarithms:

$$
\log_x(y^x) = 10 \quad \text{and} \quad \log_y(x^{4y}) = 10
$$

Our goal is to find the value of $ xy $, where $ x $ and $ y $ are
both greater than 1.

---

### Step 1:  Use the Power Rule of Logarithms

[...  solution continues ...]
Thus, the result is consistent across all derivations.

---

### Final Answer

$$
\boxed{25}
$$
```

Figure 8: Base Qwen3-8B thinking-on rollout on an AIME24 problem generated with temperature 1.0. The chat template opens a <think> block, the model reasons in the first person, tests several numerical values of x before finding a one-line argument, and closes the block with a single matched </think> before writing a polished solution. Exactly one <think> and one </think> appear in the generation. The generation is $5,067$ tokens.

Problem: There exist real numbers x and y, both greater than 1, such that $\log_x\left(y^x\right) = \log_y\left(x^{4y}\right) = 10$. Find xy.

Model output:

```
Not sure what to do with this one.  Maybe I can try to find the value
of x and y?

Okay, the problem says there exist real numbers $ x $ and $ y $, both
greater than 1, such that:
[...  reasoning continues ...]

Wait, that seems straightforward.  Did I do everything right?
Let me check again.

[...  reasoning continues ...]

This is a transcendental equation and may not have an algebraic
solution, but the problem says such real numbers exist, so it's
sufficient that we found $ xy = 25 $, and the problem doesn't ask us
to find specific x and y.

Therefore, my conclusion is that $ xy = 25 $.

So the answer should be \boxed{25}

**Final Answer**
\boxed{25}
</think>

To solve the problem, we are given two expressions involving
logarithms:

$$ \log_x(y^x) = \log_y(x^{4y}) = 10 $$

We are to determine the value of the product $ xy $, where both $ x $
and $ y $ are real numbers greater than 1.

---

### Step 1:  Simplify the First Logarithmic Expression

We begin by simplifying the logarithmic expression:

$$ \log_x(y^x) $$

Using the logarithmic identity:

$$ \log_b(a^n) = n \cdot \log_b(a) $$

[...  solution continues ...]

### Step 5:  Final Answer

We conclude that the product $ xy $ is:

$$ \boxed{25} $$
```

Figure 9: Qwen3-8B emits an orphan `</think>` token under thinking-off evaluation. The model was trained with OPSD using a thinking-mode teacher and matched privileged information z and this rollout was generated from checkpoint 160 on an AIME24 problem, taking $2,529$ tokens total. Although thinking is disabled (the chat template already supplies an empty `<think></think>` block; see Appendix C), the response opens with an extended first-person reasoning trace, boxes an answer, emits a second, unmatched `</think>`, and only then writes a polished solution, reproducing the structure of a thinking-on response.

D.5 OPSD MASKING RESULTS

To isolate LLM reasoning and grading improvements from unrelated distillation in, we do ablations for Qwen3-8B confining losses to the first 90% of the student rollout and last 10% of the student rollout.

	Thinking-on eval				Thinking-off eval			
Priv. info	**AIME24**	**AIME25**	**HMMT25**	**Avg.**	**AIME24**	**AIME25**	**HMMT25**	**Avg.**
Last 10% of tokens								
Ref. z	77.2±0.3	69.9±1.5	46.9±1.3	64.7	33.6±1.7	25.1±1.6	15.1±0.8	24.6
No z	**77.8±1.4**	69.3±1.4	**48.1±1.2**	65.1	34.8±0.9	28.4±1.6	16.9±1.0	26.7
Shuffled z	76.6±1.0	71.9±0.9	47.6±1.0	65.4	31.9±0.9	23.8±1.6	16.2±2.4	24.0
All but last 10%								
Ref. z	77.1±0.5	70.8±2.1	47.6±1.8	65.2	35.1±1.2	27.6±1.0	17.3±1.7	26.7
No z	77.7±0.6	**72.6±0.9**	47.5±0.7	**65.9**	**47.4±1.8**	**37.3±1.5**	**26.2±1.2**	**36.9**
Shuffled z	77.4±1.4	71.6±1.1	46.2±1.1	65.1	46.0±2.7	35.4±1.6	22.6±2.5	34.7

Table 17: Masking ablation for Qwen3-8B OPSD training runs evaluated with `avg@12` on AIME24, AIME25, and HMMT25. Blocks refer to which parts of student tokens the distillation loss is applied to (the rest is masked out). Rows within a block are the privileged information z shown to the teacher (the reference solution, one shuffled from another problem, or absent). Teacher thinking was on during OPSD training throughout. Each row contains the numbers aggregated over 4 seeds of a single checkpoint trained using the given privileged information. Accuracy is mean±std across those seeds. All settings and hyperparameters except masking follow (Zhao et al., 2026a).

D.6 GRADIENT ALIGNMENT RESULTS

We report gradient cosine similarities for the remaining Qwen3 models in Table 18 and for Olmo-3-7B-Instruct in Table 19, following Appendix E.3. Tables 20 and 21 report the gradient noise scale of every gradient.

Table 18: Cosine similarity between the gradient of each method trained on OpenThoughts and the $\text{GRPO}_{\text{nothink}}$ gradient on AIME24, AIME25, and HMMT25, measured at the base model (pooled over four seeds, ± paired jackknife error). Teacher thinking-on/off denotes whether thinking was enabled for the OPSD teacher, and z denotes the reference solution shown to the teacher.

Method	**Qwen3-1.7B**	**Qwen3-4B**	**Qwen3-8B**	**Qwen3-14B**
Baselines				
$\text{GRPO}_{\text{think}}$	-0.008 ± 0.013	$+0.008 \pm 0.017$	-0.014 ± 0.019	-0.004 ± 0.027
$\text{GRPO}_{\text{nothink}}$	$+0.018 \pm 0.047$	$+0.060 \pm 0.050$	$+0.118 \pm 0.050$	$+0.166 \pm 0.029$
SFT	-0.153 ± 0.026	-0.056 ± 0.021	-0.075 ± 0.031	-0.056 ± 0.015
OPSD (teacher thinking-on)				
Ref z	-0.205 ± 0.058	-0.132 ± 0.056	-0.160 ± 0.020	-0.075 ± 0.017
No z	-0.204 ± 0.062	-0.134 ± 0.041	-0.157 ± 0.019	-0.071 ± 0.018
Shuffled z	-0.198 ± 0.057	-0.139 ± 0.052	-0.157 ± 0.019	-0.074 ± 0.017
OPSD (teacher thinking-off)				
Ref z	-0.185 ± 0.058	-0.086 ± 0.056	-0.163 ± 0.013	-0.090 ± 0.023
No z	-0.087 ± 0.068	-0.145 ± 0.064	-0.172 ± 0.018	-0.087 ± 0.025
Shuffled z	-0.157 ± 0.060	-0.076 ± 0.053	-0.163 ± 0.011	-0.091 ± 0.022

Table 19: Cosine similarity between the gradient of each method trained on OpenThoughts and the GRPO gradient on AIME24, AIME25, and HMMT25 for Olmo-3-7B-Instruct, measured at the base model (pooled over four seeds, $\pm$ paired jackknife error). OPSD self-distill uses the frozen Olmo-3-7B-Instruct as the teacher, while OPSD cross-model uses Olmo-3-7B-Think as the teacher.

Method	**Cosine**
Baselines	
GRPO	$+0.026 \pm 0.021$
SFT	-0.060 ± 0.021
OPSD (self-distill)	
Ref z	$+0.057 \pm 0.052$
No z	$+0.051 \pm 0.047$
Shuffled z	$+0.057 \pm 0.052$
OPSD (cross-model)	
Ref z	$+0.052 \pm 0.031$
No z	$+0.056 \pm 0.028$
Shuffled z	$+0.052 \pm 0.031$

Table 20: Gradient noise scale across four seeds for the math probe, where each seed's gradient is the mean over 64 iterations; 0 is a noiseless estimate. The reference is $\text{GRPO}_{\text{nothink}}$ on AIME24, AIME25, and HMMT25. For Olmo-3-7B-Instruct, the teacher thinking-on and thinking-off rows are replaced by OPSD cross-model (Olmo-3-7B-Think teacher) and self-distill, respectively.

Gradient	**Qwen3-1.7B**	**Qwen3-4B**	**Qwen3-8B**	**Qwen3-14B**	**Olmo-3-7B-Instruct**
Reference	3.583	3.436	3.666	3.736	4.033
Baselines					
$\text{GRPO}_{\text{think}}$	3.979	4.108	3.889	3.786	–
$\text{GRPO}_{\text{nothink}}$	3.653	3.483	3.322	2.717	3.951
SFT	0.000	0.000	0.000	0.001	0.002
OPSD (teacher thinking-on / cross-model)					
Ref z	0.001	0.001	0.002	0.001	0.008
No z	0.002	0.002	0.003	0.002	0.011
Shuffled z	0.001	0.001	0.001	0.001	0.008
OPSD (teacher thinking-off / self-distill)					
Ref z	0.003	0.004	0.003	0.004	0.010
No z	0.010	0.008	0.010	0.007	0.017
Shuffled z	0.004	0.004	0.002	0.004	0.010

Table 21: Gradient noise scale across four seeds for the LiveCodeBench v6 probe, where each seed's gradient is the mean over 8 iterations; 0 is a noiseless estimate. The reference is GRPO on TEST_UNSEEN; GRPO and SDPO are trained on TRAIN_SEEN.

Gradient	**Qwen3-8B**	**Qwen3-14B**	**Olmo-3-7B-Instruct**
Reference	1.968	3.152	1.999
GRPO	2.855	3.125	2.076
SDPO	0.043	0.018	0.020

E IMPLEMENTATION DETAILS

All our experiments are run were run with `8xH100` nodes (with no multi-node parallelism), primarily leveraging the vLLM inference engine (Kwon et al., 2023) (we use HuggingFace Transformers (Wolf et al., 2020) for inference on SFT experiments in Section 4.2). We run our experiments on top of the OPSD codebase written in TRL (von Werra et al., 2020) and the SDPO codebase which uses veRL (Sheng et al., 2024). Our SDPO LCBv6 runs train for $\sim$40 hours, while the SDPO science and tool use jobs took $\sim$3 hours. OPSD runs took up to $\sim$2 hours and GRPO runs on the OpenThoughts dataset took up to $\sim$6 hours. OPSD thinking-on evaluations took up to $\sim$2.5 hours per dataset and OPSD thinking-off evaluations took up to $\sim$1 hour per dataset. Our gradient similarity analyses took up to $\sim$2 hours per training setting.

E.1 SDPO HYPERPARAMETERS

LiveCodeBench v6. We summarize hyperparameters used for in the LCBv6 experiments in Table 22, mirroring the hyperparameters taken from the SDPO setup (Hübotter et al., 2026). We use the exact same rich feedback settings and prompts as Hübotter et al. (2026), changing only the dataset splits.

Table 22: SDPO and GRPO training hyperparameters on LiveCodeBench v6.

Parameter	**GRPO**	**SDPO**
General		
Model	{Qwen3-8/14B, Olmo-3-7B-Instruct}	{Qwen3-8/14B, Olmo-3-7B-Instruct}
Thinking	False	False
Total steps	200	200
Data		
Max. prompt length	2048	2048
Batching		
Question batch size	32	32
Mini batch size	8	1
Number of rollouts	8	8
Rollout		
Inference engine	vLLM	vLLM
Temperature	1.0	1.0
Max. response length	8192	8192
Validation		
Number of rollouts	4	4
Temperature	0.6	0.6
Top-p	0.95	0.95
Max. response length	8192	8192
SDPO loss		
Top-K distillation	-	20
Distillation divergence	-	Reverse-KL
Clip advantages	-	-
Teacher-EMA update rate	-	0.01
Rollout importance sampling clip	-	2.0
GRPO loss		
ϵ-high	0.28	-
Rollout importance sampling clip	2.0	-
KL coefficient (λ)	0.0	-
Training		
Optimizer	AdamW	AdamW
Learning rate	1×10^{-6}	1×10^{-6}
Warmup steps	0	0
Weight decay	0.01	0.01
Gradient clip norm	1.0	1.0

SciKnowEval and ToolAlpaca. We also follow the SDPO and GRPO hyperparameters used in Hübotter et al. (2026) for SciKnowEval and ToolAlpaca, except that we find that GRPO is unstable with learning rate 1×10^{-5}, we report results with 1×10^{-6}, running SDPO and GRPO in an on-policy manner. We summarize these hyperparameters in Table 23.

Table 23: GRPO, SDPO, and SFT training hyperparameters on SciKnowEval and ToolAlpaca

Parameter	GRPO	SDPO	SFT
General			
Model	Qwen3-8B	Qwen3-8B	Qwen3-8B
Thinking	False	False	False
Total steps	200	200	200
Data			
Max. prompt length	2048	2048	2048
Batching			
Question batch size	32	32	32
Mini batch size	32	32	-
Number of rollouts	8	8	-
Rollout			
Inference engine	vLLM	vLLM	Transformers
Temperature	1.0	1.0	-
Max. response length	8192	8192	-
Validation			
Number of rollouts	16	16	16
Temperature	0.6	0.6	0.6
Top-p	0.95	0.95	0.95
Max. response length	8192	8192	8192
SDPO loss			
Top-K distillation	-	100	-
Distillation divergence	-	Jensen-Shannon	-
Clip advantages	-	-	-
Teacher-EMA update rate	-	0.05	-
Rollout importance sampling clip	-	2.0	-
Training			
Optimizer	AdamW	AdamW	AdamW
Learning rate	1×10^{-6}	1×10^{-5}	1×10^{-5}
Warmup steps	0	0	0
Weight decay	0.01	0.01	0.01
Gradient clip norm	1.0	1.0	1.0

E.2 OPSD HYPERPARAMETERS

We report our hyperparameters in Table 24; we follow the training configurations provided by Zhao et al. (2026a), training on OpenThoughts (Guha et al., 2025), changing only the content of z. The one exception is training length: we run OPSD and SFT for 100 steps longer than Zhao et al. (2026a) to evaluate fully converged performance.

E.3 GRADIENT SIMILARITY ANALYSIS

We record each method's gradients by running its training loop from the base model with learning rate zero, so the weights never change, and averaging the gradients over all iterations of each run for downstream computation. We use the hyperparameters of Tables 22 and 24, with two exceptions: GRPO uses `num_iterations`$= 1$, since at learning rate zero a second pass over the same rollouts gives the same gradient, and we record gradients with respect to all parameters rather than the LoRA adapters (for the math results), since LoRA's zero-initialized matrices may lead to poorly defined gradient behavior. We run 64 iterations per seed for OPSD on OpenThoughts and 8 for SDPO on LiveCodeBench v6, with four seeds per method.

We define the following quantities for use in our analysis:

Pooled cosine similarity. Given $S = 4$ seeds, let $\bar{g}_s$ denote the gradient of seed s averaged over its iterations, and $\bar{g} = \frac{1}{S}\sum_{s=1}^{S} \bar{g}_s$ the pooled gradient. We report the cosine similarity between the

Table 24: GRPO, OPSD, and SFT training hyperparameters on OpenThoughts. †Olmo-3-7B-Instruct has no thinking toggle, so thinking settings do not apply: it is trained and evaluated in its native mode, and its OPSD teacher is either the frozen student (self-distillation) or Olmo-3-7B-Think (cross-model).

Parameter	GRPO	OPSD	SFT
General			
Model	Qwen3-{1.7, 4, 8, 14}B, Olmo-3-7B-Instruct		
Student-thinking	{True, False}†	False†	False†
Teacher-thinking	-	{True, False}†	-
Total steps	500	200	200
Data			
Max. prompt length	2048	2048	2048
Batching			
Question batch size	4	32	32
Number of rollouts	8	1	-
Optimizer steps per generation	2	1	-
Rollout			
Inference engine	vLLM	vLLM	vLLM
Temperature	1.2	1.1	-
Top-p	1.0	0.95	-
Top-k	-	20	-
Max. response length	16000	1024	-
Validation			
Student-thinking	{True, False}†	{True, False}†	{True, False}†
Number of rollouts	12	12	12
Temperature	1.0	1.0	1.0
Top-p	0.95 (thinking-on), 0.8 (thinking-off), 0.8 (Olmo-3-7B-Instruct)		
Top-k	-	-	-
Max. response length	38912	38912	38912
OPSD loss			
Teacher policy	-	Frozen student or Olmo-3-7B-Think†	-
Teacher z	-	{ref z, no z, shuffled z}	-
Distillation divergence	-	Forward KL	-
Token clip (Qwen3-1.7B, 4B)	-	0.05	-
Token clip (Qwen3-8B, 14B)	-	0.06	-
Token clip (Olmo-3-7B-Instruct)	-	0.06	-
Training			
Optimizer	AdamW	AdamW	AdamW
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}
Warmup steps	0	0	0
Weight decay	0	0	0
Gradient clip norm	1.0	0.1	1.0
LoRA rank	64	64	64
LoRA alpha	128	128	128

pooled gradients of the reference and of each method:

$$c = \frac{\bar{g}^{\text{ref}} \cdot \bar{g}^{\text{method}}}{\|\bar{g}^{\text{ref}}\|_2 \, \|\bar{g}^{\text{method}}\|_2}. \tag{5}$$

Jackknife error. Let c_{-s} denote the pooled cosine similarity computed with seed s removed from both the reference and the method, and $\bar{c}$ the mean of the c_{-s}. The jackknife error is then

$$\text{SE} = \sqrt{\frac{S-1}{S} \sum_{s=1}^{S} (c_{-s} - \bar{c})^2}. \tag{6}$$

Gradient noise scale. Following McCandlish et al. (2018), we measure the noise of each pooled gradient as

$$\frac{\text{tr}(\Sigma)}{\|\bar{g}\|_2^2}, \tag{7}$$

where Σ is the covariance of $\bar{g}_1, \ldots, \bar{g}_S$. It is 0 for a noiseless estimate.

E.4 OPSD TEACHER AND STUDENT PROMPTS

We use the standard prompts used by Zhao et al. (2026a).

Student Prompt

```
Problem: {problem}

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

Figure 10: Student prompt for OPSD.

Privileged Teacher Prompt

```
Problem: {problem}

Here is a reference solution to this problem:
=== Reference Solution Begin ===
{z}
=== 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{}.
```

Figure 11: Teacher prompt template for OPSD. The privileged information ablations "ref z", "no z", and "shuffled z" differ only by the content of what "$\{z\}$" is replaced by in the template.