Rethinking On-Policy Self-Distillation for Thinking Models

Simran Kaur [1] **Narutatsu Ri** [1] **Yinghui He** [1] **Liam Fowl** [1] **Sanjeev Arora** [1]

Abstract

Self-distillation has emerged as a promising recipe for self-improvement in language models (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026). In this setting, a model can be used as its *own* teacher when augmented with privileged information (e.g. a solution to a math problem). This seems like an especially appealing approach for thinking models that can leverage test-time reasoning to integrate learnings from privileged information. However, we show that privileged self-distillation degrades the long-budget test-time compute behavior of thinking models: across five Qwen3 and OLMo thinking models evaluated on AIME24, AIME25, and HMMT25, privileged-context distillation causes a relative drop of up to 17% in avg@16 accuracy. The degradation scales with the amount of privileged context withheld from the student and is most pronounced at long rollout budgets, where thinking models otherwise obtain their largest gains. This failure mode is not specific to self-distillation: on-policy distillation (OPD) improves thinking models, but privileged on-policy distillation reverses these gains. Our diagnostics suggest that this failure mode is linked to how privileged teacher context reshapes learning at high-entropy forking positions (Bigelow et al., 2024; Zhang et al., 2026), i.e., rollout positions where multiple continuations remain plausible and may lead to different reasoning paths. Privileged context lowers fork rates in thinking-model rollouts but not in instruction model rollouts. This leads to an interesting dichotomy wherein privileged context can help instruction-tuned models but hurts more performant thinking models that depend heavily on exploration and rollout quality. This effect is especially visible when the student begins a self-correction branch, where privileged OPD penalizes sampled reconsideration tokens that vanilla OPD supports. Thinking models trained with a privileged teacher produce fewer verification, backtracking, and hedging markers, even after length normalization. These findings indicate that applying self-distillation methods to strong thinking models requires further consideration of token-level signal—especially around tokens related to correction and crucial reasoning steps.

[1]Princeton University, Princeton, NJ, US. Correspondence to: Simran Kaur <skaur@princeton.edu>.

Published as a paper at the 1st FoGen workshop, ICML 2026, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

On-policy self-distillation (OPSD) has emerged as an exciting approach for self-improvement in language models (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026). In this setting, a single model plays the role of both a student and teacher. The teacher is provided additional privileged information, such as a gold solution, a final answer, or environmental feedback.

Thinking models are natural candidates for self-improvement. Post-trained to deliberate at test time (OpenAI, 2024; DeepSeek-AI et al., 2025; Yang et al., 2025), they can branch into cases, verify intermediate steps, hedge, backtrack, and recover from errors before committing to an answer (Arora & Zanette, 2025; Gandhi et al., 2025; Venhoff et al., 2025). Yet existing methods have been studied outside this regime, using instruction-tuned models, short generation budgets, or trained on rollouts with thinking disabled (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026). This leaves open whether privileged-context self-distillation remains beneficial when the supervised trajectory is the long deliberation trace that thinking models rely on at test time.

In this paper, we report a negative result: existing privileged-context on-policy self-distillation methods can degrade thinking models, particularly at long rollout budgets. This degradation is not explained by the short training completion budget alone: under the same short-budget on-policy training setup, vanilla OPD with a larger teacher improves the thinking student, whereas providing the teacher with privileged context reverses the gain. Our analysis suggests a plausible explanation, namely that privileged context may suppress the deliberative behaviors that thinking models rely on at test time.

We establish this failure mode through five linked observations. First, OPSD helps instruction-tuned models more reliably than thinking models. Second, under full-solution privileged context, OPSD degrades five OpenThoughts-trained thinking models across Qwen3 and OLMo. Third, the harm scales with how much privileged context the teacher receives: final-answer-only context causes milder damage, while full-solution context causes substantially more. Fourth, Appendix K shows that the gold-demo-conditioned teacher itself does not show the same degradation: it produces much shorter rollouts while preserving high pass@k, whereas the trained student inherits shorter responses without the same longer-budget benefits. Fifth, the degradation is strongest at long rollout budgets and coincides with reduced fork rates, weaker self-correction signal, and fewer deliberation markers in trained students.

Such findings called for mechanistic exploration at the token level. The explanation appears to be a crucial sign reversal in the per-token distillation signal at high-entropy decision points, suggesting that access to privileged context suppresses exploration –e.g., branching, reconsideration, and uncertainty-marking moves. Thinking-model rollouts contain many such positions, which we call *forking positions* in line with recent work (Bigelow et al., 2024; Lin et al., 2024; Vassoyan et al., 2025; Zhang et al., 2026). They are often marked lexically by tokens such as *wait*, *hmm*, *but*, and *maybe*. Under vanilla on-policy distillation, these tokens carry positive advantage. Once privileged context is added, their advantage flips negative, and the trained student produces fewer of them at evaluation, even after length normalization.

We refer to this phenomenon as fork suppression: privileged-context self-distillation undermines the very deliberative behaviors that made thinking models natural candidates for self-improvement in the first place.

Section 2 describes the experimental setup; Section 3 establishes the empirical pattern; Section 4 analyzes the per-token signal at forking positions. We provide a discussion of related work in Appendix A, and discuss implications for self-improvement of thinking models in Appendix L.

2. Experimental Setup

We evaluate whether privileged-context on-policy self-distillation preserves the test-time search behavior of thinking models. We write the on-policy distillation objective in a context-explicit form, since our experiments vary precisely what additional information the teacher receives. Given an input problem x, the student policy π_S samples an on-policy rollout $y \sim \pi_S(\cdot \mid x)$. We then train the student by minimizing a token-level divergence between the teacher and student next-token distributions along this sampled rollout:

$$\mathcal{L}_{\text{OPD}} = \sum_t D\left(\pi_T(\cdot \mid x, c, y_{<t}), \pi_S(\cdot \mid x, y_{<t})\right).$$

Here $y_{<t}$ is the prefix before token t, π_T is the teacher policy, π_S is the student policy being updated, and c denotes teacher-only privileged context. The divergence D can be instantiated as forward KL, reverse KL, JSD, or another token-level distillation divergence. Vanilla OPD corresponds to $c = \emptyset$, while privileged-context OPD sets c to additional information such as a final answer or gold demonstration. In OPSD, the teacher and student share the same architecture and typically start from the same checkpoint; in our experiments, the teacher is initialized as a copy of the initial student checkpoint and receives privileged context while the student does not. Unless otherwise noted, training rollouts are capped at 4,096 completion tokens. We use JSD distillation for one epoch with effective batch size 64; this training cap is shared by the vanilla OPD, privileged OPD, and OPSD comparisons. Full hyperparameter details are in Appendix B.

Training data. We use two training domains. For math reasoning, we train on an OpenThoughts 15k subset, using the problem as the student prompt and the full reference solution as the privileged teacher context. For Countdown, we train on 15k examples from `jasonrqh/Countdown-CoT-20k`; the privileged context is the post-thinking solution suffix. The Countdown split also reserves 500 held-out examples for in-domain evaluation.

Models and comparisons. Our main comparisons separate instruct models from thinking models. On Countdown, we train paired instruct and thinking models from the Qwen3-4B (Yang et al., 2025) and OLMo-3-7B (Team Olmo et al., 2025) families to test whether OPSD behaves differently when the base model already performs explicit deliberation. On OpenThoughts, we focus on thinking models across sizes and families, including Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Qwen3-4B-Think-2507, and OLMo-7B-Think. We also include controls that compare OPSD to vanilla OPD and that disable thinking during OPSD training while keeping thinking enabled at evaluation.

Evaluation. We evaluate transfer to AIME 2024, AIME 2025, and HMMT 2025, and evaluate Countdown-trained models on the held-out Countdown split. For AIME/HMMT, each benchmark has 30 problems. To reflect standard evaluation practice for thinking models, which often uses long rollout budgets to benefit from test-time compute scaling, our main evaluation adopts a long generation cap: we generate 16 samples per problem, each with a maximum length of 38,912 tokens. This evaluation cap is much larger than

Table 1. **OPSD degrades thinking models across model families.** We train each thinking model with privileged OpenThoughts solution context and evaluate on AIME24, AIME25, and HMMT25. Entries report avg@16 accuracy, and the Average column averages the three benchmarks. OPSD lowers the average score for all five thinking models, suggesting that the degradation is not specific to one model size or family.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B (Thinking)	**Base**	**0.502**	**0.398**	**0.215**	**0.372**
	+OPSD	0.435	0.302	0.185	0.308
Qwen3-4B (Thinking)	**Base**	**0.727**	**0.635**	**0.410**	**0.591**
	+OPSD	0.683	0.556	0.362	0.534
Qwen3-8B (Thinking)	**Base**	**0.758**	**0.700**	**0.448**	**0.635**
	+OPSD	0.721	0.613	0.400	0.578
Qwen3-4B-Think-2507	**Base**	**0.804**	**0.804**	**0.552**	**0.720**
	+OPSD	0.787	0.731	0.529	0.683
OLMo-7B-Think	**Base**	**0.719**	**0.667**	**0.452**	**0.612**
	+OPSD	0.715	0.652	0.446	0.604

Table 2. **Privileged teacher context, not on-policy distillation itself, degrades thinking-model performance.** On a Qwen3-1.7B thinking student, vanilla OPD with a larger Qwen3-8B teacher improves avg@16 accuracy over the base model. However, conditioning the teacher on information unavailable to the student at test time makes the resulting on-policy update harmful on average. This occurs both for a larger teacher (OPD gold demo) and for the self-teacher (OPSD) setting. All OPD/OPSD variants are trained with a 4,096-token completion cap; all methods, including the base model, are evaluated with a 38,912-token generation cap.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B	**Base**	0.502	**0.398**	0.215	0.372
	+OPSD	0.435	0.302	0.185	0.308
	+OPD	**0.540**	0.385	**0.252**	**0.392**
	+OPD gold demo	0.467	0.344	0.240	0.350

the default 4,096-token training completion cap. Main tables report avg@16 accuracy, the mean sample correctness over the 16 rollouts for each problem. Note that avg@16 is distinct from unbiased pass@16, which measures whether at least one of the 16 samples solves the problem. To compare the main long-budget setting with shorter evaluation settings, we repeat evaluation at generation caps of 4,096, 8,192, 16,384, 32,768, and 38,912 tokens, and also report pass@1[1] and unbiased pass@16.

3. Results

3.1. OPSD Helps Instruct Models, Can Degrade Thinking Models

OPSD training helps instruct models more reliably than thinking models. In the Countdown-trained setting (Table 21), Qwen3-4B-Instruct-2507 and OLMo-7B-Instruct improve under OPSD ($0.527 \rightarrow 0.559$ and $0.474 \rightarrow 0.494$ avg@16 performance). The matched thinking models show mixed results: Qwen3-4B-Think-2507 drops slightly ($0.776 \rightarrow 0.765$), while OLMo-7B-Think gains modestly ($0.679 \rightarrow 0.695$). Appendix F shows a similar pattern in an SD-Zero-style self-revision pipeline (He et al., 2026): the self-revision stage helps both the instruction-tuned and thinking models, but the OPSD stage further helps only the instruction-tuned model and hurts the thinking model.

Thinking model degradation is clearest in the OpenThoughts-trained setting (Table 1). Across model scale and family, all five thinking models we evaluate lose avg@16 performance under the same training recipe, with drops ranging from 6.4 points (Qwen3-1.7B) to 0.8 points (OLMo-7B-Think). Paired problem-level bootstrap intervals for the main average deltas are reported in Appendix C.1; in this setting, the intervals exclude zero for four of the five comparisons, including all Qwen variants.

For thinking models, OPSD degrades performance when the rollouts being supervised are thinking rollouts. Table 22 holds the model fixed (Qwen3-4B) and evaluates with thinking enabled in all rows. When training rollouts are non-thinking, performance is preserved ($0.591 \rightarrow 0.590$); when they are thinking, performance drops to 0.532.

[1]Empirical pass@1 is the same per-sample correctness quantity as avg@16, estimated from sampled rollouts under the evaluation decoding distribution.

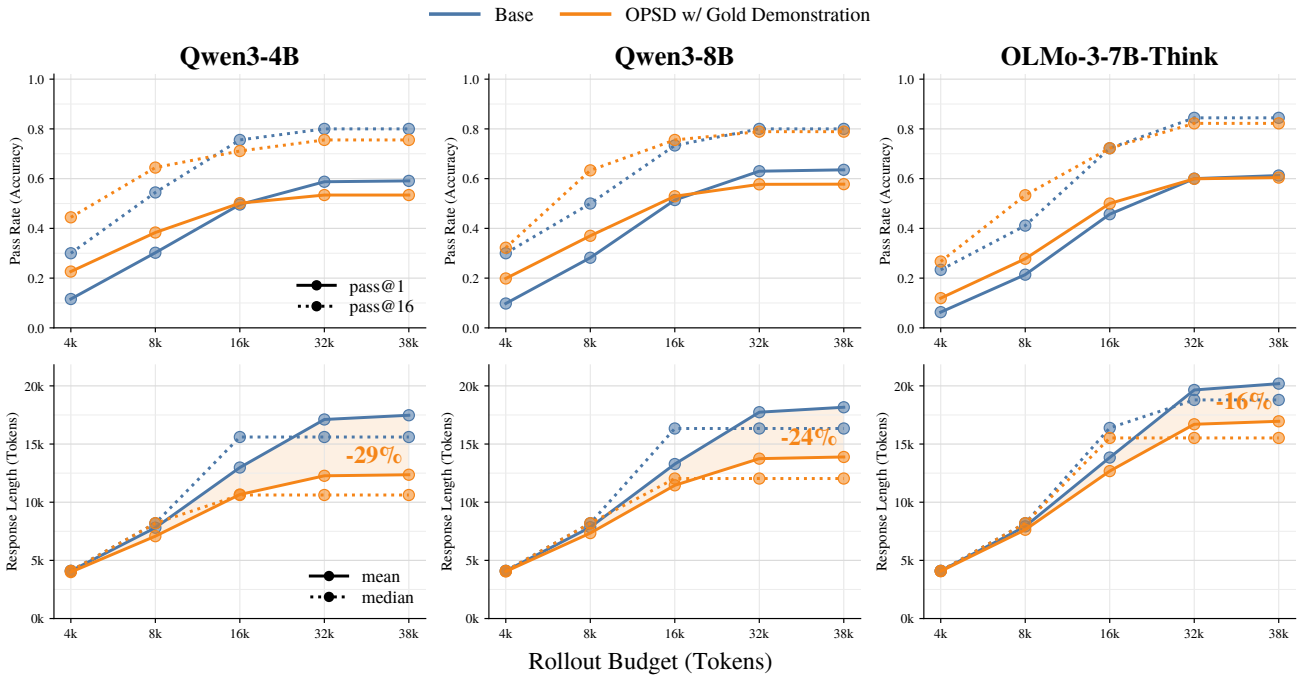

Figure 1. **For thinking models, OPSD can improve short-budget performance via compression but can hurt long-budget reasoning.** We evaluate OpenThoughts-trained Qwen3-4B, Qwen3-8B, and OLMo-3-7B thinking models across rollout budgets. Models are trained with a 4,096-token completion cap and evaluated at caps from 4,096 to 38,912 tokens. Top row: pass@1 and pass@16, averaged over AIME24, AIME25, and HMMT25. Bottom row: mean and median response length. OPSD with gold demonstrations often helps at 4k–8k tokens, but the advantage shrinks or reverses at 32k–38k tokens, where responses become substantially shorter than the base model.

3.2. Degradation Is Specific to Privileged-Context OPSD

The same student improves under vanilla on-policy distillation, where the teacher is not given privileged context. In Table 2, vanilla OPD trains Qwen3-1.7B against a larger teacher that sees the same prompt, improving avg@16 performance from 0.372 to 0.392. Adding privileged context reverses the sign: context-enhanced OPD with a privileged gold demonstration reduces performance to 0.350, and OPSD with a full gold demonstration reduces it to 0.308. Performance falls when the teacher scores the student's trajectory while conditioned on information the student will not have at test time.

3.3. Degradation Appears With Longer Thinking

The degradation in thinking-model performance is concentrated at long rollout budgets. Figure 1 evaluates OpenThoughts-trained Qwen3-4B, Qwen3-8B, and OLMo-7B-Think across rollout budgets from 4k to 38k tokens. At 4k–8k tokens, OPSD models perform comparably to or above their bases[2]. By 32k–38k tokens, they match or fall below. Response length follows the same pattern: at small budgets, base and OPSD rollouts are similar in length; at large budgets, OPSD rollouts are substantially shorter. Overall, OPSD removes the gains that thinking models obtain from longer rollouts.

One might suspect a simpler explanation based on the mismatch between training and evaluation lengths. Training rollouts are capped at 4,096 tokens, whereas evaluation allows longer rollouts, so short-budget training may cause the model to forget reasoning behaviors. The OPD experiments rule out this explanation as the sole cause. In Table 2, vanilla OPD with a larger unprivileged teacher improves the student under the same short-budget on-policy setup, whereas adding teacher-only gold-demonstration context reverses the gain. The corresponding budget curves in Figure 6 show that the two OPD variants differ not only in accuracy but also in response length: vanilla OPD largely preserves the base response-length behavior, while gold-demonstration OPD shortens responses and reduces pass@1 gains. Thus, the drop cannot be attributed only to the train/eval budget mismatch; it also depends on whether the teacher scores the short-budget rollout with privileged context.

3.4. Sparse Privileged Context Better Preserves Long-Budget Behavior

We next ask whether the student-side degradation depends on how much privileged context the teacher receives, and

[2]Throughout this section, "base" denotes the corresponding instruct or thinking checkpoint before additional OPSD/OPD training, not a pretrained base model.

observe that full gold demonstrations degrade long-budget behavior more than final-answer-only context. Figure 8 compares these two forms of privileged context, averaged over the models in Table 1. Full demonstrations give the largest short-budget gains in pass@1 and pass@16, but also the largest long-budget reversals, with mean rollout length compressed to roughly $0.8\times$ the base at 38k tokens. The final-answer-only condition gives smaller short-budget gains, remains closer to the base at long budgets, and produces less length compression. Because the answer-only and full-demonstration runs use the same training recipe and 4,096-token completion cap, this difference cannot be explained by the short training budget alone; it depends on how much privileged context the teacher receives.

In Section 4, we show that OPSD reduces fork rates at high-entropy decision points and reduces explicit deliberation markers in evaluation rollouts.

4. Privileged Context Reduces Forking

Section 3 isolates when privileged-context distillation damages thinking models: the degradation appears most clearly at long rollout budgets and grows with the density of the teacher's privileged context. We next analyze a possible mechanism: privileged context changes the per-token distillation signal on forking tokens, which mark uncertainty or redirection, and trained students produce fewer such tokens at evaluation. The evidence has three parts. First, privileged context shifts the fork–lock structure of thinking-model rollouts (§4.1). Second, at these positions, the token-level distillation signal reverses on self-correction and uncertainty markers (§4.2). Third, trained OPSD students produce fewer deliberation markers at evaluation (§4.3).

4.1. Privileged Context Shifts Fork-Lock Distribution

Following the SSD diagnostic of Zhang et al. (2026), we distinguish *fork* positions from *lock* positions. A fork is a high-entropy position where several plausible continuations remain available and may lead to different reasoning paths; a lock is a locally constrained position where the continuation is comparatively determined. We classify positions from the teacher distribution along sampled rollouts with a top-candidate dominance heuristic: a position is a fork when the top candidate has probability below 0.25, and a lock when the top candidate has probability above 0.65. We then report per-rollout fork and lock rates.

Figure 2 compares Qwen3-4B-Instruct-2507 and Qwen3-4B with thinking enabled under three teacher conditions: no privileged context, sparse context containing the gold final answer, and dense context containing the full gold demonstration. The two modes respond differently. For the instruction-tuned model, fork rates remain near 0.06 and

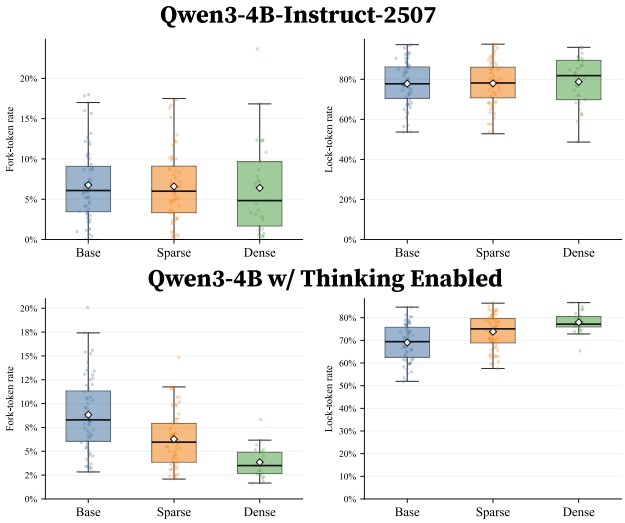

Figure 2. **Dense privileged context lowers fork rates in thinking rollouts but has little effect on instruction-style rollouts.** We compute fork and lock rates using the SSD-style diagnostic of Zhang et al. (2026): fork positions are high-entropy decision points with multiple plausible continuations, while lock positions are locally determined continuations. Panels show the OPSD side of the diagnostic for base, sparse privileged-context, and dense privileged-context conditions; the left column shows fork rate and the right column shows lock rate. For the instruction-tuned model, fork and lock rates remain mostly flat as privileged context becomes denser, with at most a small dense-context reduction in fork rate. For the thinking model, denser privileged context lowers fork rate and raises lock rate, consistent with the hypothesis that OPSD suppresses the branching points needed for test-time search.

lock rates near 0.78, essentially flat across conditions. For the thinking model, denser privileged context monotonically lowers fork rates and raises lock rates: the median fork rate drops from roughly 0.083 to 0.035 between the base and dense conditions, while the median lock rate rises from roughly 0.69 to 0.77.

Privileged context reshapes the rollout structure of thinking models but leaves instruction-style rollouts essentially unchanged.

4.2. Privileged Context Reverses Signal on Fork Markers

The fork-rate changes in §4.1 align with a sign reversal in the per-token distillation signal. Forks are often marked by short epistemic and revision tokens such as *wait*, *hmm*, *but*, and *maybe*. These markers are lexical proxies for forking positions, not direct measurements. We inspect the per-token teacher–student log-ratio along sampled rollouts: positive values mean the teacher assigns more probability than the student to the sampled token, and negative values mean the teacher assigns less.

Figure 3 visualizes the reversal. The trajectory is held fixed

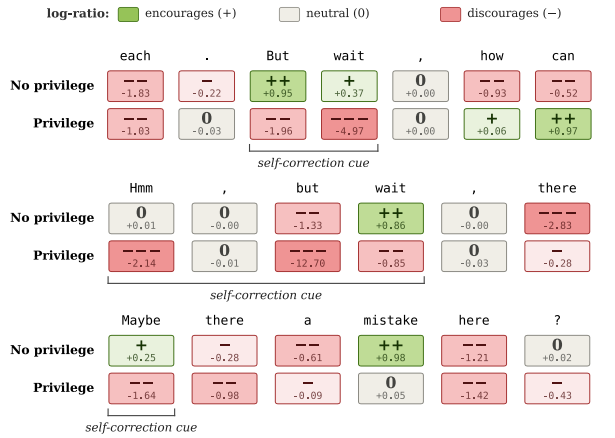

Figure 3. **Privilege flips credit on self-correction cues, even when they lead to the correct answer.** Three windows from rollouts of a Qwen3-1.7B (Thinking) student trained against a Qwen3-8B (Thinking) teacher on OpenThoughts; the trajectory is identical under both teachers; only the teacher differs. Cells show the sign and magnitude of the per-token log-ratio $\log \pi_T(y_t \mid y_{<t}, x) - \log \pi_S(y_t \mid y_{<t}, x)$. **Top:** a rollout for *"radius of a circle tangent to two concentric circles"* that ends at the wrong answer; privilege assigns negative advantage to `But` `wait` (-1.96, -4.97) and rewards the dismissive continuation `how` `can`. **Middle and bottom:** two windows from a rollout that reaches the *correct* answer. Privilege still suppresses every self-correction cue, most extremely `but` (-12.70) and the hedge `Maybe` (-1.64). The pattern is consistent: the privileged teacher's positive credit moves away from the moments where the student notices something is off and onto the locally fluent continuation.

and scored by two teachers: an unprivileged teacher that sees the same prompt as the student, and a privileged teacher that also sees answer information. In a rollout that ends at the wrong answer, the privileged teacher strongly suppresses the self-correction cue `But` `wait` (-1.96, -4.97) and shifts positive credit toward the locally fluent continuation `how` `can`. The same pattern appears in windows from a rollout that reaches the correct answer: the privileged teacher suppresses self-correction cues such as `but` (-12.70) and `Maybe` (-1.64), even though the resulting trajectory reaches the right answer.

Table 24 reports the realized density of the same marker set in Qwen3-1.7B evaluation rollouts. Vanilla OPD leaves the aggregate epistemic-token density nearly unchanged ($1.080\% \rightarrow 1.074\%$), while the gold-demonstration variant lowers it to 0.850%. The reduction appears on several individual markers, including *wait* ($3.85 \rightarrow 2.54$ occurrences per 1,000 generated tokens) and *hmm* ($1.45 \rightarrow 1.11$).

The same pattern appears before sampling, in the probability mass assigned to epistemic markers: vanilla OPD leaves marginal marker mass essentially unchanged, while the gold-demonstration teacher lowers it, with the largest drops on *wait*, *recall*, *altern*, and *hmm*. Full numbers and token-mask controls are in Appendix E.

4.3. Trained Student Produces Fewer Deliberation Markers

The token-level analysis suggests that students trained with privileged context should produce fewer deliberation markers at evaluation. We test this by comparing paired base and OPSD rollouts with the same model, benchmark, problem, and sample index (see Appendix B.6). The analysis uses the five thinking models in the OpenThoughts 15k comparison on AIME24, AIME25, and HMMT25, giving 7,200 paired rollouts.

Table 23 counts three families of deliberation markers: verification markers such as *check* and *verify*, backtracking markers such as *wait*, *actually*, and *wrong*, and hedging markers such as *maybe* and *seems*. OPSD reduces the raw count of all three marker families, and the reduction survives length normalization. Verification markers drop from 1.63 to 1.35 per 1,000 generated tokens (95% CI $[-0.32, -0.25]$), with smaller but still negative shifts in backtracking and hedging. The length collapse of Section 3 is therefore accompanied by a within-rollout reduction in deliberation markers, not just shorter rollouts containing them at the same density.

Appendix F gives complementary evidence from SD-Zero: the self-revision stage alone helps the thinking model, but the subsequent OPSD stage reverses that gain.

These diagnostics do not establish a causal explanation for the accuracy drop. Rather, they identify a consistent behavioral pattern associated with the drop. Privileged-context distillation degrades long-budget thinking behavior; the degradation grows with the density of privileged teacher context; privileged context lowers fork rates and raises lock rates in thinking-model rollouts; fixed-trajectory scoring shows sign reversals on self-correction cues; and trained students emit fewer deliberation markers. Together, these observations support our interpretation of fork suppression as a failure mode of privileged token-level supervision in thinking models, while leaving open whether it is the primary cause of the observed accuracy degradation.

Accessibility

We have prepared this submission with accessibility in mind, providing descriptive figure and table captions, and avoiding reliance on color alone to convey information.

Software and Data

This work uses publicly available datasets and model checkpoints summarized in Appendix B.2, and the standard inference engine vLLM. We do not redistribute third-party datasets or model checkpoints. Anonymized code and supporting artifacts will be released upon de-anonymization.

Impact Statement

This paper presents work whose goal is to advance the understanding of self-distillation methods for language models that perform extended test-time reasoning. Our results clarify when privileged-context self-distillation can degrade thinking models, which we hope informs more robust self-improvement methods. There are many potential societal consequences of advances in language-model reasoning, none of which we feel must be specifically highlighted here beyond the general considerations that apply to research on capable language models.

References

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Arora, D. and Zanette, A. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AiZxn84Wdo.

Bigelow, E., Holtzman, A., Tanaka, H., and Ullman, T. Forking paths in neural text generation. *arXiv preprint arXiv:2412.07961*, 2024.

Cursor Team. Introducing composer 2.5. https://cursor.com/blog/composer-2-5, May 2026.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Ding, K. Hdpo: Hybrid distillation policy optimization via privileged self-distillation. *arXiv preprint arXiv:2603.23871*, 2026.

Gandhi, K., Chakravarthy, A. K., Singh, A., Lile, N., and Goodman, N. D. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. In *Proceedings of the 2nd Conference on Language Modeling (COLM 2025)*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819, 2021.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

He, Y., Kaur, S., Bhaskar, A., Yang, Y., Liu, J., Ri, N., Fowl, L., Panigrahi, A., Chen, D., and Arora, S. Self-distillation zero: Self-revision turns binary rewards into dense supervision, 2026. URL https://arxiv.org/abs/2604.12002.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hübotter, J., Lübeck, F., Behric, L., Baumann, A., Bagatella, M., Marta, D., Hakimi, I., Shenfeld, I., Buening, T. K., Guestrin, C., et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Kim, J., Luo, X., Kim, M., Lee, S., Kim, D., Jeon, J., Li, D., and Yang, Y. Why does self-distillation (sometimes) degrade the reasoning capability of llms?, 2026. URL https://arxiv.org/abs/2603.24472.

Kudithipudi, D., Aguilar-Simon, M., Babb, J., Bazhenov, M., Blackiston, D., Bongard, J., Brna, A. P., Chakravarthi Raja, S., Cheney, N., Clune, J., et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.

Lin, Z., Liang, T., Xu, J., Lin, Q., Wang, X., Luo, R., Shi, C., Li, S., Yang, Y., and Tu, Z. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *arXiv preprint arXiv:2411.19943*, 2024.

OpenAI. Learning to reason with LLMs, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2026-05-07.

Qu, Y., Setlur, A., Smith, V., Salakhutdinov, R., and Kumar, A. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Sang, H., Xu, Y., Zhou, Z., He, R., Wang, Z., and Sun, J. On-policy self-distillation for reasoning compression. *arXiv e-prints*, pp. arXiv–2603, 2026.

Setlur, A., Wang, Z., Cohen, A., Rashidinejad, P., and Xie, S. M. Reuse your flops: Scaling rl on hard problems by conditioning on very off-policy prefixes. *arXiv preprint arXiv:2601.18795*, 2026.

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Team Olmo, Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., Heineman, D., Groeneveld, D., Brahman, F., Timbers, F., Ivison, H., Morrison, J., Poznanski, J., Lo, K., Soldaini, L., Jordan, M., Chen, M., Noukhovitch, M., Lambert, N., Walsh, P., Dasigi, P., Berry, R., Malik, S., Shah, S., Geng, S., Arora, S., Gupta, S., Anderson, T., Xiao, T., Murray, T., Romero, T., Graf, V., Asai, A., Bhagia, A., Wettig, A., Liu, A., Rangapur, A., Anastasiades, C., Huang, C., Schwenk, D., Trivedi, H., Magnusson, I., Lochner, J., Liu, J., Miranda, L. J. V., Sap, M., Morgan, M., Schmitz, M., Guerquin, M., Wilson, M., Huff, R., Bras, R. L., Xin, R., Shao, R., Skjonsberg, S., Shen, S. Z., Li, S. S., Wilde, T., Pyatkin, V., Merrill, W., Chang, Y., Gu, Y., Zeng, Z., Sabharwal, A., Zettlemoyer, L., Koh, P. W., Farhadi, A., Smith, N. A., and Hajishirzi, H. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Vassoyan, J., Beau, N., and Plaud, R. Ignore the kl penalty! boosting exploration on critical tokens to enhance rl fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6108–6118, 2025.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Understanding reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and Planning for Large Language Models at ICLR 2025*, 2025. URL https://openreview.net/forum?id=OwhVWNOBcz.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yang, C., Qin, C., Si, Q., Chen, M., Gu, N., Yao, D., Lin, Z., Wang, W., Wang, J., and Duan, N. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

Zhang, R., Bai, R. H., Zheng, H., Jaitly, N., Collobert, R., and Zhang, Y. Embarrassingly simple self-distillation improves code generation. *arXiv preprint arXiv:2604.01193*, 2026.

Zhao, S., Xie, Z., Liu, M., Huang, J., Pang, G., Chen, F., and Grover, A. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

A. Related Work

Continual Learning. Model weights are updated during pre-training and post-training, but are then often deployed as static artifacts for months to serve traffic. This can quickly lead to a gap between in-weight knowledge and relevant real-world skills and information. *Continual learning* aims to address this very gap (De Lange et al., 2021; Kudithipudi et al., 2022; Hadsell et al., 2020; Wang et al., 2024). Recently, self-distillation methods have claimed to allow for near seamless continual learning on new tasks while avoiding catastrophic forgetting common in other continual learning approaches (Shenfeld et al., 2026). This line of work claims that the policy being updated is minimally changed when trained using on-policy distillation from a teacher shown privileged information.

On-Policy Distillation. Distillation methods aim to impart richer knowledge into a student model (usually from an already trained teacher) compared to standard supervised learning methods (Hinton et al., 2015; Gou et al., 2021; Agarwal et al., 2024). For language modeling tasks, several variants of distillation exist. *Sequence distillation* uses natural language outputs of one model to directly fine-tune a student model. *Knowledge distillation* goes one step further and trains a student using the probabilities that a teacher assigns to its generated sequence. *On-policy distillation* has emerged as an alternative to standard knowledge distillation wherein the sequence being scored, and subsequently used for training, is generated *on-policy* by the student model. This approach alleviates observed issues between train-test distribution shift that can arise in standard knowledge distillation.

Self-Improving Distillation. Generally speaking, knowledge distillation methods, including sequence distillation and on/off-policy distillation, require a stronger teacher in order to improve the student's performance. This can induce a heavy computational overhead for researchers and also raises an important question: can a model be used to improve itself without the need for a stronger teacher? Several recent approaches have attempted to answer this question in the affirmative. On-Policy Self-Distillation (OPSD) (Zhao et al., 2026) utilizes a single model architecture to act as both teacher and student: the teacher policy is conditioned on privileged information, such as a ground-truth answer to a math problem, and is used to score on-policy generations of a student without the privileged information. It is worth noting that recent updates to the OPSD method specifically disable per-token divergence feedback on deliberation tokens, which is necessary to stabilize training. Self-Distillation Fine-Tuning (SDFT) (Shenfeld et al., 2026) proposes a similar method but focuses on integrating new knowledge corpora for continual learning rather than verifiable math questions.

We focus on self-distillation methods in the previous vein: methods that use ground truth privileged information to directly condition a teacher's token-level feedback to train a student. However, other flavors of self-distillation methods may target different objectives and mechanisms. For example, CRISP (Sang et al., 2026) focuses on reducing thinking trace lengths by distilling a teacher's concise reasoning into a student model. Self-distillation has also been attempted where the teacher, instead of being given privileged information or different steering prompts, is instead sampled at different temperatures (Zhang et al., 2026). Approaches like Self-Distillation Policy Optimization (SDPO) (Hübotter et al., 2026) also use privileged information but augment the teacher policy with environmental feedback like error messages instead of gold, ground-truth information like OPSD or SDFT.

Self-Distillation Failure Modes. Concurrent work (Kim et al., 2026) studies why self-distillation can sometimes degrade mathematical reasoning, and attributes the degradation to suppression of epistemic verbalization under richer teacher conditioning. Their analysis emphasizes context richness and task coverage: richer conditioning produces shorter, more confident traces with fewer uncertainty expressions, which can help narrow in-domain settings but hurt OOD math generalization. Viewed through our framework, epistemic-verbalization suppression is a visible lexical subset of fork suppression: overt uncertainty markers expose some high-entropy fork positions, but many branch-relevant decisions occur at ordinary mathematical, connective, or formatting continuations. Our work is therefore complementary. We focus on long-rollout thinking models and ask how privileged token-level feedback changes the high-entropy decision points that support test-time search. Rather than treating epistemic markers as the primary object of study, we analyze fork-like positions in the teacher distribution, token-level signal reversal along fixed student trajectories, and the resulting loss of long-budget test-time compute gains.

Forking and Exploration in Reasoning Traces. Reasoning ability from a token-level perspective has been analyzed in several works that have found that some tokens, called forking tokens, can have an outsized influence on the downstream success of a reasoning trace (Bigelow et al., 2024; Lin et al., 2024; Vassoyan et al., 2025; Zhang et al., 2026). These critical tokens often occur at high-entropy positions in a model's generation and control where and how the reasoning branches into

different paths and alternative strategies. Our work contextualizes failures of self-improvement distillation methods in the setting of forking suppression.

Privileged Feedback in RL. Standard RL setups may face exploration bottlenecks on complex reasoning tasks where correct rollouts can be rare. To overcome this, several methods in RL also aim to incorporate privileged information into the exploration stage. For example, Privileged On-Policy Exploration (POPE) (Qu et al., 2026) and PrefixRL (Setlur et al., 2026) utilize some privileged information to guide on-policy exploration. Hybrid approaches like HDPO (Ding, 2026) augment standard RL training with targeted privileged self-distillation in certain cases. Similar frameworks like Self-Distilled RLVR (RLSD) (Yang et al., 2026) incorporate self-distillation to guide update magnitudes of standard RLVR training. Our work sets aside RL training mechanisms to focus on the token-level dynamics of privileged distillation—a harm that hybrid approaches may be able to mitigate.

B. Experimental Details

This appendix records the data, training, and evaluation settings used for the experiments in the main text. Unless a table or paragraph states otherwise, all reported OPSD runs use the defaults in Table 5 and all evaluations use the generation and grading protocol in Table 7.

B.1. Data

OpenThoughts math. We use a cleaned version of `Ashkchamp/Openthoughts_math_filtered_30K`. The cleaning script removes system turns, remaps thought delimiters to `<think>` and `</think>`, removes explicit solution delimiters, and appends the instruction to return the final answer in `\boxed{}`. We use a 15K subset of the data for training. The privileged teacher context is taken from the `solution` column.

Countdown. For Countdown, we use `jasonrqh/Countdown-CoT-20k`. We select a 15K subset for training and reserve an additional 500 examples as a held-out evaluation set.

Table 3. Training and evaluation data used in the experiments.

Data	Rows	Use and fields
OpenThoughts cleaned	29,439	Source pool after cleaning; columns include `problem`, `solution`, and `Answer`.
OpenThoughts 15k	15,000	Main OpenThoughts OPSD training subset; prompt = `problem`, privileged context = `solution`.
Countdown train	15,000	Countdown OPSD training; prompt = `datapoint_input_text`, privileged context = `response_suffix`.
Countdown eval	500	Held-out Countdown evaluation.
AIME24/AIME25/HMMT25	30 each	Math competition evaluations.

B.2. Existing Asset Licenses

Table 4 summarizes the existing datasets, model checkpoints, and software assets used in our experiments. We use these assets for training, evaluation, or implementation, and do not redistribute third-party datasets or model checkpoints.

Table 4. Existing assets used in this work.

Asset	Source	License / terms	Use
`Ashkchamp/Openthoughts_math_filtered_30K`	https://huggingface.co/datasets/Ashkchamp/Openthoughts_math_filtered_30K	No explicit license metadata listed on the dataset page at time of access	Source pool for the OpenThoughts math training subset; not redistributed.
`open-thoughts/OpenThoughts-114k`	https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k	Apache-2.0	Upstream OpenThoughts dataset related to the math training data; not redistributed.
`jasonrqh/Countdown-CoT-20k`	https://huggingface.co/datasets/jasonrqh/Countdown-CoT-20k	MIT	Countdown training and held-out in-domain evaluation data.
Qwen3 checkpoints	https://huggingface.co/Qwen	Apache-2.0	Base instruction and thinking models used for training and evaluation.
OLMo-3 checkpoints	https://huggingface.co/allenai/Olmo-3-7B-Think	Apache-2.0; Ai2 Responsible Use Guidelines	Base instruction and thinking models used for training and evaluation.
`idanshen/Self-Distillation`	https://github.com/idanshen/Self-Distillation	No explicit software license file found at time of access	Reference/adapted implementation code for self-distillation; not redistributed.
`vLLM`	https://github.com/vllm-project/vllm	Apache-2.0	Inference engine used for generation and evaluation.
AIME and HMMT problems	Public competition materials	Public competition materials; not redistributed	Out-of-domain math evaluation benchmarks.

B.3. Prompt Templates and Privileged Context Examples

We fill the following templates before applying each model's chat template. For standard OPSD training, the student receives only the problem.

Student prompt template

```
{problem}
```

The privileged teacher receives the same problem plus an example response.

Privileged-teacher prompt template

```
{problem}

This is an example for a response to the question:

{Answer}

Now answer with a response of your own, including the thinking process.
```

Here `{Answer}` denotes the privileged response used as teacher context. In dense gold-demonstration runs, this field contains a full reference solution; in sparse final-answer-only runs, it contains only the boxed final answer.

For the conciseness-control experiment in Figure 7, the teacher is prompted without gold context.

Conciseness-control teacher prompt

```
Solve the following math problem concisely and correctly.  Be direct -- avoid
unnecessary elaboration, redundant steps, or restating the problem.  Focus only on
the key reasoning steps needed to reach the answer.

{problem}
```

Sparse and dense context example. The dense and sparse conditions use the same privileged-teacher wrapper above; they differ only in what is inserted into the {Answer} slot. The following filled examples use the same problem.

Dense teacher prompt, filled

Given real numbers a, b, c and a positive number λ such that the polynomial $f(x) = x^3 + ax^2 + bx + c$ has three real roots x_1, x_2, x_3, and the conditions $x_2 - x_1 = \lambda$ and $x_3 > \frac{1}{2}(x_1 + x_2)$ are satisfied, find the maximum value of

$$\frac{2a^3 + 27c - 9ab}{\lambda^3}.$$

This is an example for a response to the question:

We begin by analyzing the function $f(x) = x^3 + ax^2 + bx + c$, which has three real roots x_1, x_2, x_3. We are given the following conditions: $x_2 - x_1 = \lambda$ and $x_3 > \frac{1}{2}(x_1 + x_2)$. We aim to find the maximum value of

$$\frac{2a^3 + 27c - 9ab}{\lambda^3}.$$

Transform the polynomial to remove the quadratic term. Substitute $x = y - \frac{a}{3}$ into $f(x)$:

$$\begin{aligned} F(y) &= f\left(y - \tfrac{a}{3}\right) \\ &= \left(y - \tfrac{a}{3}\right)^3 + a\left(y - \tfrac{a}{3}\right)^2 + b\left(y - \tfrac{a}{3}\right) + c \\ &= y^3 - \left(\tfrac{a^2}{3} - b\right) y + \tfrac{1}{27}(2a^3 + 27c - 9ab). \end{aligned}$$

Identify the new roots of $F(y)$. Let the roots of $F(y)$ be y_1, y_2, y_3. We know $y_i = x_i + \frac{a}{3}$. Using Vieta's formulas,

$$y_1 + y_2 + y_3 = 0, \qquad y_1 y_2 y_3 = -\frac{1}{27}(2a^3 + 27c - 9ab).$$

[Middle of the gold demonstration omitted for clarity of the example.]

Conclusion:

$$\boxed{\frac{3\sqrt{3}}{2}}$$

Now answer with a response of your own, including the thinking process.

Sparse final-answer-only teacher prompt, filled

Given real numbers a, b, c and a positive number λ such that the polynomial $f(x) = x^3 + ax^2 + bx + c$ has three real roots x_1, x_2, x_3, and the conditions $x_2 - x_1 = \lambda$ and $x_3 > \frac{1}{2}(x_1 + x_2)$ are satisfied, find the maximum value of

$$\frac{2a^3 + 27c - 9ab}{\lambda^3}.$$

This is an example for a response to the question:

$$\boxed{\frac{3\sqrt{3}}{2}}$$

Now answer with a response of your own, including the thinking process.

B.4. OPSD Training

For each training example, the student is prompted with the task input alone. The teacher is initialized from the same base checkpoint, but receives the same task input plus privileged supervision. The trainer samples completions on-policy and minimizes token-level divergence between teacher and student distributions on those sampled completion tokens.

Table 5. Default OPSD training hyperparameters.

Hyperparameter	Value
Objective	Generalized KL/JSD distillation on sampled completion tokens
Distillation mixture α	0.5 (JSD)
Teacher synchronization	Disabled for reported regular runs
Completions per prompt	1
Training sampling temperature / top-p	1.0 / 1.0
Epochs	1
Effective batch size	64
Per-device batch size	1
Gradient accumulation	8
Optimizer learning-rate schedule	Cosine decay
Warmup ratio	0.1
Max gradient norm	1.0
Weight decay	0.0
Precision	bf16
Distributed training	FSDP full-shard auto-wrap
Gradient checkpointing	Enabled
Seed	42
Max prompt length	25,000 tokens
Max completion length	4,096 tokens unless noted otherwise

Table 6. Model-specific OPSD training settings and deviations from Table 5.

Model(s)	Training data	LR	Max completion	vLLM train gen.	Hardware
Qwen3-1.7B, Qwen3-4B, Qwen3-4B-Thinking-2507, Qwen3-4B-Instruct-2507	OpenThoughts 15k	5×10^{-6}	4,096	Yes	8 H100, 80GB/GPU
Qwen3-8B	OpenThoughts 15k	2×10^{-6}	4,096	Yes	8 H100, 80GB/GPU
OLMo-3-7B-Instruct, OLMo-3-7B-Think	OpenThoughts 15k	5×10^{-6}	4,096	Yes	8 H100, 80GB/GPU
Qwen3-4B-Thinking-2507, Qwen3-4B-Instruct-2507, OLMo-3-7B-Instruct, OLMo-3-7B-Think	Countdown 15k	5×10^{-6}	4,096	Yes	8 H100, 80GB/GPU

B.5. Evaluation

For AIME24, AIME25, and HMMT25, we generate 16 samples per problem on 30 problems per benchmark. Generation is sharded over 8 jobs and uses a maximum generation length of 38,912 tokens. The merged generation file therefore contains 480 rows for each AIME/HMMT benchmark. Countdown uses the same 16-sample evaluation protocol on the 500-example held-out split when reported.

Table 7. Evaluation generation hyperparameters.

Model/eval group	Temperature	Top-p
Qwen3 thinking-style models	0.6	0.95
Qwen3-4B-Instruct-2507	0.7	0.8
OLMo-3-7B-Instruct/Think	0.6	0.95

Metrics. The reported avg@16 `accuracy` is the mean correctness over generated samples: for each problem, we average correctness across its 16 sampled rollouts, then average across problems. This is equivalent to empirical single-sample correctness under the evaluation sampling distribution, but is distinct from unbiased pass@16. For $k > 1$, pass@k is computed with the unbiased estimator. For each problem with n samples and c correct samples, the contribution is 1 if $n - c < k$ and otherwise

$$1 - \prod_{i=0}^{k-1} \frac{n-c-i}{n-i}.$$

We report pass@k only for $k \leq n$.

Table 8. Shared evaluation settings.

Setting	Value
Samples per problem	16
Shards	8
Maximum generation length	38,912 tokens
Eval engine	vLLM
Eval precision	bf16
vLLM GPU memory utilization	0.9
Eval top-k	-1
Prompt format	Model chat template with generation prompt
Metric file	`metrics.json` written next to merged JSONL

Table 9. **Full deliberation-marker counts for paired base and OPSD rollouts.** Raw columns report average marker counts per response. Normalized columns report occurrences per 1,000 generated tokens, using response-token counts computed with the cached model tokenizer. Deltas are OPSD minus base, with confidence intervals computed by clustered bootstrap over model–benchmark–problem clusters.

Marker family	**Base raw**	**OPSD raw**	**Raw Δ**	**Base /1k**	**OPSD /1k**	**Δ /1k**
Verification	26.1	17.3	-8.8 [-9.5, -8.1]	1.63	1.35	-0.28 [-0.32, -0.25]
Backtracking	124.2	100.6	-23.5 [-29.0, -19.1]	6.45	6.29	-0.16 [-0.32, -0.02]
Hedging	72.2	55.2	-17.0 [-18.5, -15.5]	3.79	3.62	-0.17 [-0.25, -0.10]

B.6. Deliberation Marker Analysis

To test whether OPSD changes explicit deliberation language in model reasoning, we compare paired base and OPSD rollouts for the same model, benchmark, problem, and sample (see Table 9). The analysis uses the five thinking models in the OpenThoughts 15k comparison on AIME24, AIME25, and HMMT25. With 16 samples for each of 30 problems on each benchmark, this gives 7,200 paired rollouts.

For each response, we count occurrences of a fixed, case-insensitive lexicon of deliberation markers. The marker families are verification markers, with examples such as *check*, *verify*, *double check*, and *make sure*; backtracking markers, with examples such as *wait*, *actually*, *mistake*, *wrong*, and *another way*; and hedging markers, with examples such as *maybe*, *probably*, *might*, and *seems*. We report raw marker counts and marker counts per 1,000 response tokens. Response-token counts are computed with the cached model tokenizer.

Deltas are OPSD minus base. Confidence intervals are computed by clustered bootstrap over model–benchmark–problem clusters, so the 16 samples from the same problem are not treated as fully independent. The mean response length in this analysis is 19,395 tokens for base rollouts and 15,391 tokens after OPSD. Since the counts are lexical proxies, we interpret them as evidence about explicit deliberation markers in the generated traces, not as direct measurements of latent uncertainty or confidence.

C. Accuracy Confidence Intervals and Full Pass@k Results

C.1. Paired Bootstrap Confidence Intervals

Tables 10, 11, and 12 report paired 95% bootstrap confidence intervals for the avg@16 deltas in Tables 21, 1, and 2, respectively. For this metric, we first compute each problem's contribution as the mean correctness over its 16 sampled rollouts. For each bootstrap replicate, we resample evaluation problems with replacement within each benchmark, preserving the paired measurements for the two methods being compared. We then compute the method delta within each benchmark and average benchmark deltas using the same aggregation as the corresponding table. Intervals are percentile intervals from 10,000 bootstrap replicates. These intervals reflect evaluation-problem uncertainty for the observed 16-sample estimates, but not training-seed variability. Thus, for a 30-problem benchmark, the bootstrap operates over 30 paired problem-level observations, not 480 rollout-level observations.

Table 10. **OPSD helps instruct models more reliably than thinking models on Countdown.** Companion to Table 21; Δ rows report paired bootstrap 95% confidence intervals for avg@16 deltas.

Model		Countdown	AIME24	AIME25	HMMT25	Average
Qwen3-4B-Instruct-2507	**Base**	0.736	0.604	0.463	0.304	0.527
	+OPSD	0.865	0.594	0.483	0.294	0.559
	Δ (OPSD - Base)	$+0.129$	-0.010	$+0.021$	-0.010	$+0.032$
	Δ [95% CI]	$[0.117, 0.141]$	$[-0.052, 0.033]$	$[-0.010, 0.054]$	$[-0.044, 0.019]$	$[0.017, 0.048]$
Qwen3-4B-Think-2507	**Base**	0.945	0.804	0.804	0.552	0.776
	+OPSD	0.947	0.800	0.775	0.537	0.765
	Δ (OPSD - Base)	$+0.002$	-0.004	-0.029	-0.015	-0.011
	Δ [95% CI]	$[-0.010, 0.014]$	$[-0.027, 0.019]$	$[-0.058, -0.002]$	$[-0.062, 0.021]$	$[-0.026, 0.002]$
OLMo-7B-Instruct	**Base**	0.719	0.525	0.415	0.237	0.474
	+OPSD	0.814	0.510	0.394	0.256	0.494
	Δ (OPSD - Base)	$+0.095$	-0.015	-0.021	$+0.019$	$+0.020$
	Δ [95% CI]	$[0.083, 0.107]$	$[-0.056, 0.025]$	$[-0.058, 0.013]$	$[-0.002, 0.040]$	$[0.005, 0.034]$
OLMo-7B-Think	**Base**	0.877	0.719	0.667	0.452	0.679
	+OPSD	0.890	0.742	0.698	0.452	0.695
	Δ (OPSD - Base)	$+0.013$	$+0.023$	$+0.031$	0.000	$+0.017$
	Δ [95% CI]	$[0.006, 0.021]$	$[-0.006, 0.052]$	$[0.002, 0.067]$	$[-0.025, 0.031]$	$[0.004, 0.031]$

Table 11. **OPSD degrades thinking models across model families.** Companion to Table 1; Δ rows report paired bootstrap 95% confidence intervals for avg@16 deltas.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B (Thinking)	**Base**	0.502	0.398	0.215	0.372
	+OPSD	0.435	0.302	0.185	0.308
	Δ (OPSD - Base)	-0.067	-0.096	-0.029	-0.064
	Δ [95% CI]	$[-0.125, -0.017]$	$[-0.160, -0.042]$	$[-0.062, 0.002]$	$[-0.095, -0.037]$
Qwen3-4B (Thinking)	**Base**	0.727	0.635	0.410	0.591
	+OPSD	0.683	0.556	0.362	0.534
	Δ (OPSD - Base)	-0.044	-0.079	-0.048	-0.057
	Δ [95% CI]	$[-0.100, 0.004]$	$[-0.138, -0.019]$	$[-0.098, -0.002]$	$[-0.088, -0.027]$
Qwen3-8B (Thinking)	**Base**	0.758	0.700	0.448	0.635
	+OPSD	0.721	0.613	0.400	0.578
	Δ (OPSD - Base)	-0.037	-0.087	-0.048	-0.058
	Δ [95% CI]	$[-0.085, 0.006]$	$[-0.160, -0.027]$	$[-0.098, 0.000]$	$[-0.090, -0.028]$
Qwen3-4B-Think-2507	**Base**	0.804	0.804	0.552	0.720
	+OPSD	0.787	0.731	0.529	0.683
	Δ (OPSD - Base)	-0.017	-0.073	-0.023	-0.037
	Δ [95% CI]	$[-0.052, 0.017]$	$[-0.121, -0.029]$	$[-0.081, 0.037]$	$[-0.065, -0.010]$
OLMo-7B-Think	**Base**	0.719	0.667	0.452	0.612
	+OPSD	0.715	0.652	0.446	0.604
	Δ (OPSD - Base)	-0.004	-0.015	-0.006	-0.008
	Δ [95% CI]	$[-0.033, 0.023]$	$[-0.058, 0.025]$	$[-0.037, 0.027]$	$[-0.028, 0.011]$

C.2. Full Pass@k Results

The main text reports avg@16 values for readability. Tables in this appendix give the corresponding full-format versions of the main result tables, with each benchmark cell written as pass@1 / pass@16. The pass@1 values are identical to

Table 12. **Privileged teacher context, not on-policy distillation itself, degrades thinking-model performance.** Companion to Table 2; Δ rows report paired bootstrap 95% confidence intervals for avg@16 deltas.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B	**Base**	0.502	0.398	0.215	0.372
	+OPSD	0.435	0.302	0.185	0.308
	+OPD	0.540	0.385	0.252	0.392
	+OPD gold demo	0.467	0.344	0.240	0.350
	Δ **(OPD - Base)**	$+0.037$	-0.013	$+0.037$	$+0.021$
	Δ [95% CI]	$[-0.006, 0.081]$	$[-0.056, 0.027]$	$[0.013, 0.065]$	$[-0.001, 0.042]$
	Δ **(OPD gold demo - OPD)**	-0.073	-0.042	-0.013	-0.042
	Δ [95% CI]	$[-0.123, -0.023]$	$[-0.079, -0.008]$	$[-0.044, 0.019]$	$[-0.065, -0.019]$
	Δ **(OPSD - Base)**	-0.067	-0.096	-0.029	-0.064
	Δ [95% CI]	$[-0.125, -0.017]$	$[-0.160, -0.042]$	$[-0.062, 0.002]$	$[-0.095, -0.037]$

the corresponding main-table avg@16 values, because both are computed as mean correctness over the same 16 sampled rollouts per problem. The pass@16 values instead estimate the probability that at least one of 16 sampled rollouts solves the problem, using the unbiased estimator described in Appendix B.5.

Table 13. **Full pass@1/pass@16 results for the Countdown-trained think-vs-instruct comparison.** This is the full-format companion to Table 21. Entries report pass@1 / pass@16 on held-out Countdown data, AIME24, AIME25, and HMMT25. The Average column averages the four benchmarks.

Model		Countdown	AIME24	AIME25	HMMT25	Average
Qwen3-4B-Instruct-2507	**Base**	0.736 / 0.964	**0.604** / 0.833	0.463 / 0.733	**0.304 / 0.500**	0.527 / 0.758
	+OPSD	**0.865 / 0.968**	0.594 / **0.867**	**0.483 / 0.767**	0.294 / **0.500**	**0.559 / 0.775**
Qwen3-4B-Think-2507	**Base**	0.945 / **0.996**	**0.804 / 0.933**	**0.804 / 0.900**	**0.552 / 0.767**	**0.776 / 0.899**
	+OPSD	**0.947 / 0.996**	0.800 / **0.933**	0.775 / **0.900**	0.537 / **0.767**	0.765 / **0.899**
OLMo-7B-Instruct	**Base**	0.719 / 0.952	**0.525 / 0.867**	**0.415** / 0.700	0.237 / 0.467	0.474 / 0.746
	+OPSD	**0.814 / 0.978**	0.510 / **0.867**	0.394 / **0.733**	**0.256 / 0.500**	**0.494 / 0.769**
OLMo-7B-Think	**Base**	0.877 / 0.996	0.719 / 0.900	0.667 / 0.833	**0.452 / 0.800**	0.679 / 0.882
	+OPSD	**0.890 / 0.998**	**0.742 / 0.933**	**0.698 / 0.867**	**0.452** / 0.733	**0.695 / 0.883**

Table 14. **Full pass@1/pass@16 results for OpenThoughts-trained thinking models.** This is the full-format companion to Table 1. Entries report pass@1 / pass@16 on AIME24, AIME25, and HMMT25. The Average column averages the three benchmarks.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B (Thinking)	**Base**	**0.502 / 0.800**	**0.398 / 0.667**	**0.215 / 0.467**	**0.372 / 0.644**
	+OPSD	0.435 / **0.800**	0.302 / 0.600	0.185 / 0.433	0.308 / 0.611
Qwen3-4B (Thinking)	**Base**	**0.727 / 0.867**	**0.635 / 0.867**	**0.410 / 0.667**	**0.591 / 0.800**
	+OPSD	0.683 / 0.833	0.556 / 0.833	0.362 / 0.600	0.534 / 0.756
Qwen3-8B (Thinking)	**Base**	**0.758** / 0.833	**0.700 / 0.833**	**0.448 / 0.733**	**0.635 / 0.800**
	+OPSD	0.721 / **0.900**	0.613 / 0.833	0.400 / 0.633	0.578 / 0.789
Qwen3-4B-Think-2507	**Base**	**0.804 / 0.933**	**0.804** / 0.900	**0.552** / 0.767	**0.720 / 0.867**
	+OPSD	0.787 / 0.867	0.731 / **0.900**	0.529 / **0.800**	0.683 / 0.856
OLMo-7B-Think	**Base**	**0.719** / 0.900	**0.667** / 0.833	**0.452 / 0.800**	**0.612 / 0.844**
	+OPSD	0.715 / **0.933**	0.652 / **0.867**	0.446 / 0.667	0.604 / 0.822

Table 15. **Full pass@1/pass@16 results for the Qwen3-1.7B OPD comparison.** This is the full-format companion to Table 2. Entries report pass@1 / pass@16 on AIME24, AIME25, and HMMT25. The Average column averages the three benchmarks.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B	**Base**	0.502 / **0.800**	**0.398** / **0.667**	0.215 / 0.467	0.372 / 0.644
	+OPSD	0.435 / **0.800**	0.302 / 0.600	0.185 / 0.433	0.308 / 0.611
	+OPD	**0.540** / **0.800**	0.385 / **0.667**	**0.252** / 0.567	**0.392** / 0.678
	+OPD gold demo	0.467 / **0.800**	0.344 / **0.667**	0.240 / **0.600**	0.350 / **0.689**

D. Fork/Lock Token Measurement

For each model family, we measured fork- and lock-like token positions by evaluating teacher next-token distributions on fixed student traces. We used the same 60 OpenMathReasoning prompts for every model and generated one student reasoning trace per prompt using the base student prompt. For each trace, we evaluated the teacher distribution at every generated token position under three conditioning settings: *base*, *sparse*, and *dense*. The base condition included only the problem statement; the sparse condition additionally provided the correct final answer; and the dense condition provided privileged context in the form of a truncated reference solution from a stronger model.

For each token position t, we formed the teacher context $(x, y_{<t})$ and stored the teacher's retained top-K next-token log-probabilities. In the final SRT runs, we used $K = 3$ and set `max_student_tokens` to 3072 for the base and sparse conditions. For dense runs, we capped the reference trace at 2048 tokens and the student trace at 1536 tokens to avoid prompt-logprob memory failures. All six cells per model were completed: OPSD and OPD crossed with base, sparse, and dense, with 60 traces per cell.

Entropy-threshold analysis. We first normalized the entropy of the retained top-K distribution:

$$H_K^{\text{norm}} = \frac{-\sum_{i=1}^{K} q_i \log q_i}{\log K},$$

where q_i denotes the top-K probabilities renormalized over the retained support. Positions with $H_K^{\text{norm}} \leq 0.20$ were labeled *lock* tokens, positions with $H_K^{\text{norm}} \geq 0.60$ were labeled *fork* tokens, and all remaining positions were labeled neutral.

Support-aware SSD approximation. We also classified positions using the geometry of the retained support. Starting from the saved top-K distribution, we applied top-p truncation with $p = 0.8$ and computed the retained support size, top-token probability, top-1/top-2 log-probability gap, entropy-derived effective support size

$$N_{\text{eff}} = \exp(H_S),$$

and the number of competitive tokens within a factor of 3 of the top token. A position was labeled lock-like when the retained support was sharply concentrated, and fork-like when multiple retained tokens remained competitive. Tokens outside the retained support were treated as tail mass rather than forks. Positions satisfying neither criterion were labeled neutral.

Aggregation. For each trace and conditioning setting, we computed fork, lock, and neutral rates as the fraction of classified token positions in the trace. We visualize per-trace rates using boxplots, separately for OPSD and OPD, with the base, sparse, and dense conditions shown in each panel. Boxes summarize the distribution across 60 traces; jittered points show individual traces; and diamond markers indicate means.

E. OPD Ablations

We ablate where the OPD loss is applied in the Qwen3-1.7B OpenThoughts comparison from Table 2. Vanilla OPD applies the unprivileged teacher's loss to all sampled response tokens. Epistemic-token OPD applies the same loss only to tokens in the epistemic-marker set. Random-fraction OPD is a token-count-matched control: if x is the average fraction of epistemic tokens in student responses, then each rollout receives OPD loss on a uniformly sampled $x\%$ subset of response tokens. OPD + privileged gold-demonstration context uses the same loss over response tokens but conditions the teacher on a gold demonstration.

Table 16. **Token-masked OPD ablations do not reproduce the gold-demonstration degradation pattern.** We evaluate Qwen3-1.7B OPD variants on AIME24, AIME25, and HMMT25. Entries report pass@1 / pass@16 to match the full-format tables in Appendix C. Epistemic-token OPD applies the loss only on epistemic-marker tokens. Random-fraction OPD applies the loss to a random fraction of response tokens matched to the average epistemic-token rate. Both token-masked OPD variants improve over the base model on average, while the gold-demonstration variant drops below the base on average at pass@1.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-1.7B	**Base**	0.502 / **0.800**	0.398 / 0.667	0.215 / 0.467	0.372 / 0.644
	Vanilla OPD	**0.540** / **0.800**	0.385 / 0.667	0.252 / 0.567	**0.392** / 0.678
	Epistemic-only OPD	0.523 / **0.800**	0.396 / 0.733	0.235 / 0.533	0.385 / 0.689
	Random-frac OPD	0.494 / **0.800**	**0.408** / **0.767**	**0.256** / 0.533	0.386 / **0.700**
	OPD + gold demo	0.467 / **0.800**	0.344 / 0.667	0.240 / **0.600**	0.350 / 0.689

Table 16 suggests that token-masked OPD can still offer some improvement over the base thinking model even when the loss is applied to only a small fraction of response tokens. The epistemic-token and random matched-fraction masks are close enough that these results do not clearly rank one mask above the other. This is consistent with the idea that lexical epistemic markers such as *wait* and *hmm* are useful proxies for forking behavior, but do not exhaust it: branch-relevant decisions can also occur on ordinary mathematical, connective, or formatting tokens. A random matched-fraction mask may therefore sample some consequential non-lexical decision points, while the epistemic-token mask targets explicit deliberation markers more directly.

Table 17. **Gold-demonstration context also lowers probability mass on epistemic tokens.** This token-level companion to Table 24 reports the model probability assigned to epistemic markers. The Marginal column gives aggregate probability mass on the epistemic-token set; the named columns give log-probabilities for representative markers; and Avg. logp averages over the set. Vanilla OPD leaves these probabilities nearly unchanged, while the gold-demonstration variant lowers both the aggregate marginal and several revision-token log-probabilities.

Method	Marginal	wait	recall	okay	altern	check	verify	hmm	Avg. logp
Base	0.00928	-0.57	-0.39	-0.15	-0.64	-0.38	-0.66	-0.72	-0.502
+OPD	0.00929	-0.57	-0.38	-0.15	-0.64	-0.38	-0.67	-0.70	-0.499
+OPD gold demo	0.00853	-0.72	-0.50	-0.17	-0.82	-0.40	-0.67	-0.96	-0.605

Table 18. **Sparse-loss OPD controls leave epistemic-marker probabilities close to vanilla OPD.** We report aggregate probability mass on the epistemic-marker set and log-probabilities for representative markers. Epistemic-token OPD and random-fraction OPD remain nearly identical to vanilla OPD on the aggregate marginal and average log-probability. Conditioning the teacher on a privileged gold demonstration lowers the marginal mass and assigns substantially lower probability to several revision markers, especially *wait*, *recall*, *altern*, and *hmm*.

Method	Marginal	wait	recall	okay	altern	check	verify	hmm	Avg. logp
Base	0.00928	-0.57	-0.39	-0.15	-0.64	-0.38	-0.66	-0.72	-0.502
Vanilla OPD	0.00929	-0.57	-0.38	-0.15	-0.64	-0.38	-0.67	-0.70	-0.499
Epistemic-only OPD	0.00928	-0.57	-0.38	-0.15	-0.64	-0.38	-0.66	-0.71	-0.498
Random-frac OPD	0.00928	-0.57	-0.37	-0.15	-0.64	-0.37	-0.67	-0.72	-0.499
OPD + gold demo	0.00853	-0.72	-0.50	-0.17	-0.82	-0.40	-0.67	-0.96	-0.605

Table 19. **Gold-demonstration context produces the largest drop in realized epistemic-token density.** The aggregate density column reports the fraction of generated tokens in the epistemic-marker set. The remaining columns report occurrences per 1,000 generated tokens for representative markers. Epistemic-token OPD and random-fraction OPD slightly reduce aggregate marker density relative to the base and vanilla OPD, but the gold-demonstration variant produces the largest decrease, including clear reductions in *wait* and *hmm*.

Method	Epistemic token density	wait	recall	okay	altern	check	verify	hmm
Base	1.080%	3.85	1.11	2.17	0.64	1.33	0.25	1.45
Vanilla OPD	1.074%	3.79	1.04	2.17	0.72	1.19	0.16	1.68
Epistemic-only OPD	1.047%	3.22	1.05	2.23	0.82	1.43	0.18	1.54
Random-frac OPD	1.039%	3.24	1.11	2.17	0.90	1.27	0.18	1.52
OPD + gold demo	0.850%	2.54	0.96	2.07	0.66	1.05	0.10	1.11

Table 20. **The OPSD stage helps an instruction-tuned model but hurts a thinking model.** We compare the base model, OPSD alone, self-revision training alone (SRT), and the full SRT+OPSD pipeline on Qwen3-4B-Instruct and Qwen3-4B. Entries report avg@8 accuracy on AIME24, AIME25, HMMT25, and their average.

Model		AIME24	AIME25	HMMT25	Average
Qwen3-4B-Instruct	**Base**	59.6	45.8	26.7	44.0
	+OPSD	63.3	47.9	32.9	48.0
	SRT	66.7	59.2	40.0	55.3
	SRT + OPSD	**68.3**	**60.0**	**45.4**	**57.9**
Qwen3-4B (Thinking)	**Base**	72.5	**65.4**	45.4	61.1
	+OPSD	63.3	60.0	44.6	56.0
	SRT	**73.3**	63.3	**50.0**	**62.2**
	SRT + OPSD	70.0	63.3	43.3	58.9

F. SD-Zero Self-Revision Pipeline

The interpretation in Section 4.3 is also consistent with an existing self-distillation pipeline in which the OPSD stage is problematic for thinking models even when surrounding stages help. SD-Zero (He et al., 2026) first trains a model to revise its own responses using reward feedback, then distills the reviser back into the generator with an on-policy self-distillation step. We compare the base, the self-revision training stage alone (SRT), and the full SRT+OPSD pipeline on Qwen3-4B-Instruct and Qwen3-4B.

Table 20 shows that the pipeline behaves as intended on the instruction-tuned model: SRT improves the base by 11.3 points ($44.0 \rightarrow 55.3$), and the OPSD stage adds another 2.6 points ($55.3 \rightarrow 57.9$). On the thinking model, SRT also helps slightly ($61.1 \rightarrow 62.2$), but the subsequent OPSD stage reverses the gain and leaves the model 3.3 points below SRT alone ($62.2 \rightarrow 58.9$). The self-revision stage is not the problem; the OPSD stage that follows it is.

G. Additional Budget-Curve Figures

Figures 4 and 5 split the budget-dependent results in Figure 1 into pass-rate and response-length views across the five thinking models in Table 1. Figure 7 adds a conciseness-prompt comparison for Qwen3-8B. Figure 6 gives the corresponding budget-curve view for the context-enhanced OPD comparison in Table 2.

Relation to CRISP-style reasoning compression. CRISP (Sang et al., 2026) studies a complementary setting in which the teacher is conditioned on a conciseness instruction rather than on a gold answer or reference solution. Thus, unlike gold-context OPSD, CRISP does not give the teacher task-answer information, but its token-level supervision can still act globally across the rollout: the teacher is encouraged to prefer shorter, more direct continuations at many positions. Our Qwen3-8B conciseness-prompt control confirms the first-order CRISP effect, namely that such conditioning shortens responses. However, the comparison with final-answer-only and full-demonstration OPSD shows that response shortening is not unique to conciseness distillation. Dense gold-demonstration context produces the strongest long-budget compression, while final-answer-only context remains closer to the base model. Thus, our claim is not that compression itself is always harmful, but that token-level teachers which broadly suppress deliberative continuations can remove the long-budget gains of thinking models.

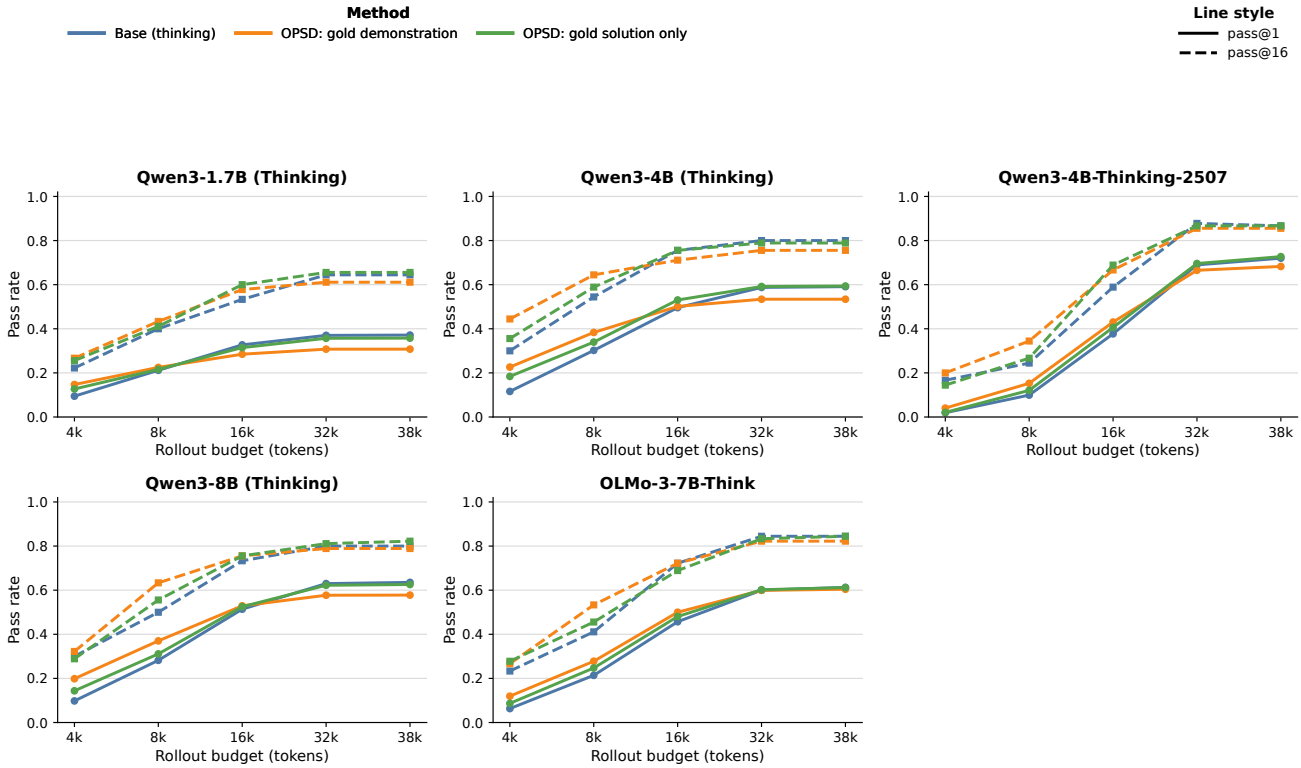

Figure 4. **Per-model pass-rate budget curves separate the accuracy effects in Figure 1.** We evaluate five OpenThoughts-trained thinking models at rollout budgets from 4k to 38k tokens on AIME24, AIME25, and HMMT25. Solid lines show pass@1 and dashed lines show pass@16. Blue curves are base thinking models, orange curves are OPSD with full gold-demonstration context, and green curves are OPSD with final-answer-only privileged context. Dense demonstrations tend to give larger short-budget gains, while the advantage narrows or reverses at longer budgets for several models.

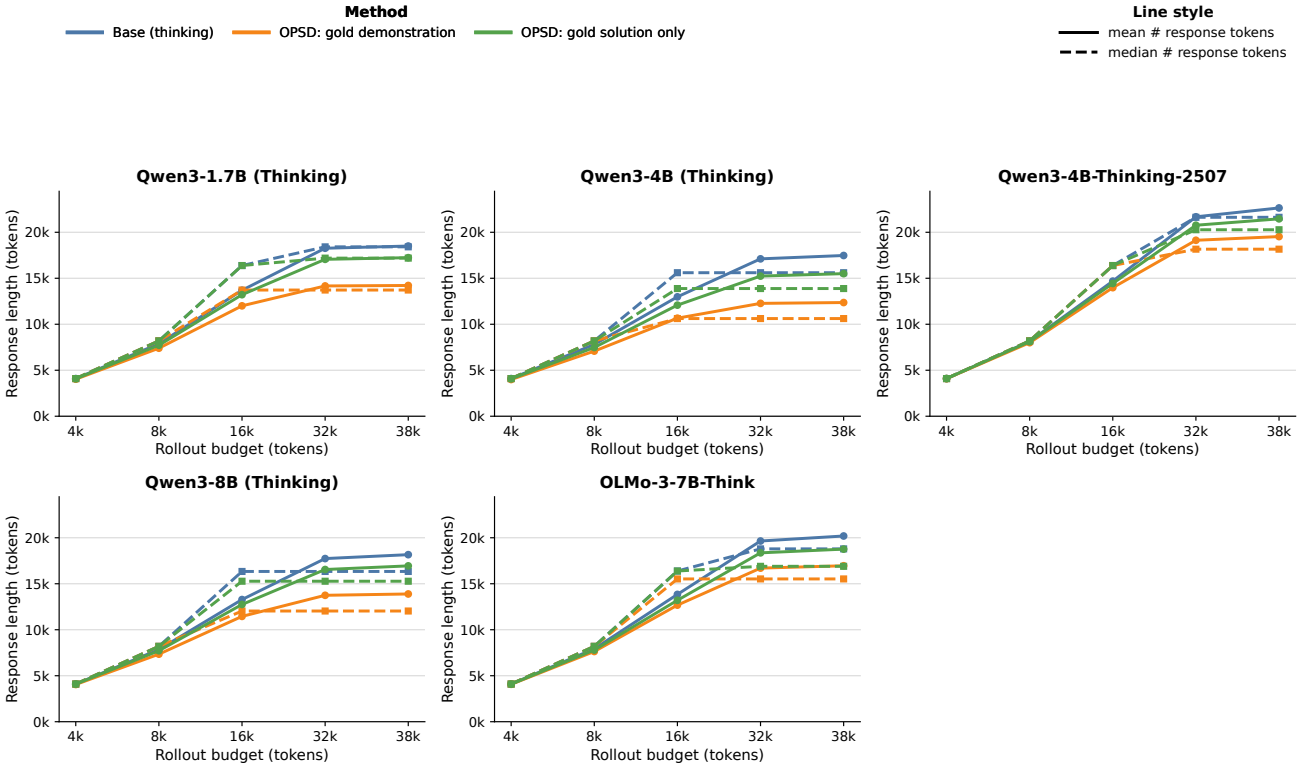

Figure 5. **Per-model response-length budget curves show where OPSD compresses long thinking rollouts.** This companion to Figure 1 reports response-token counts for the same five OpenThoughts-trained thinking models, evaluation benchmarks, and rollout budgets as Figure 4. Solid lines show mean response length and dashed lines show median response length. Blue curves are base thinking models, orange curves are OPSD with full gold-demonstration context, and green curves are OPSD with final-answer-only privileged context. At 32k–38k token budgets, full-demonstration OPSD generally produces shorter responses than the corresponding base model.

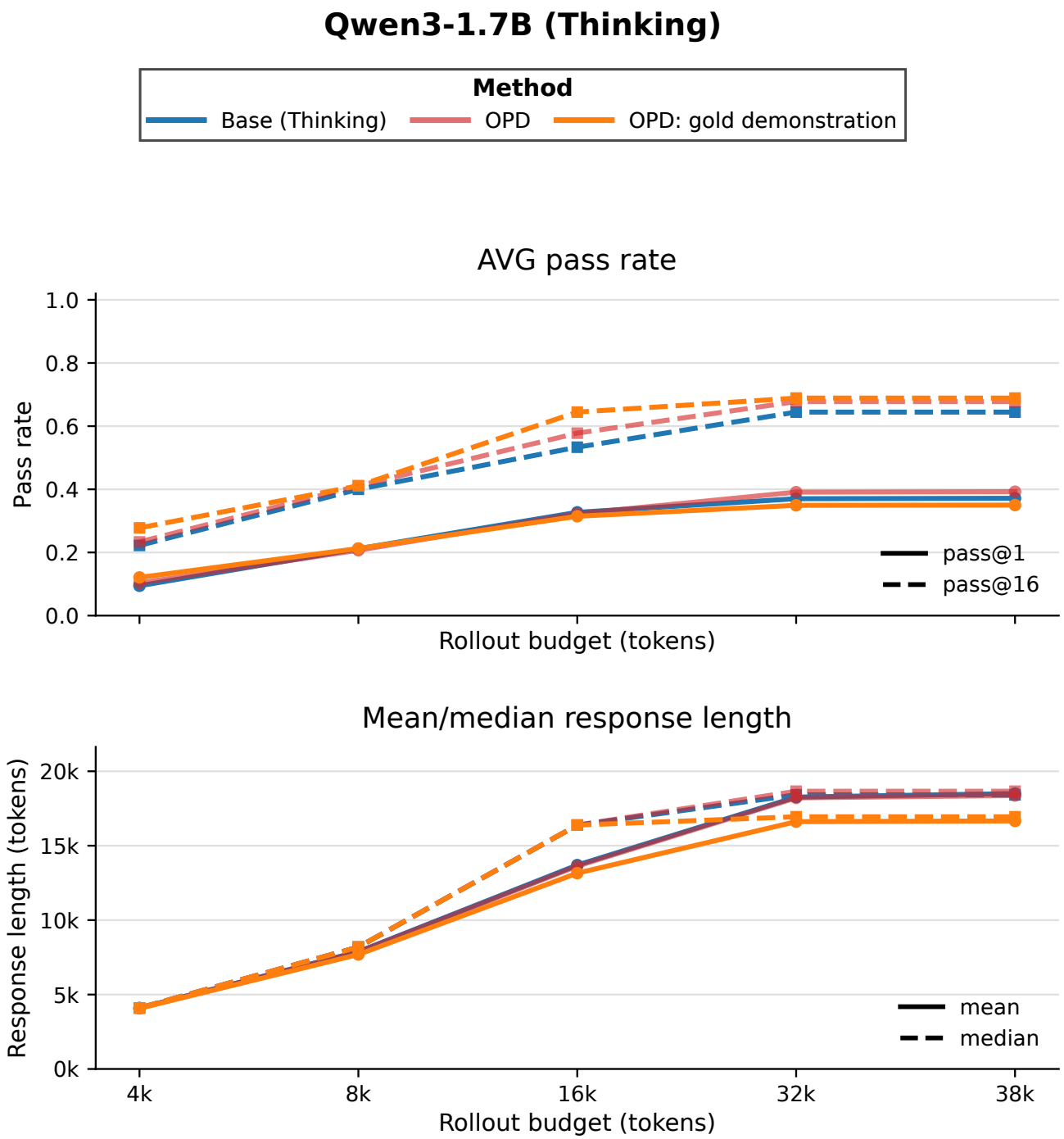

Figure 6. **OPD with gold-demonstration context shortens response lengths while vanilla OPD preserves the base length profile.** This budget-curve companion to Table 2 evaluates the Qwen3-1.7B thinking student with a larger Qwen3-8B teacher. We compare the base model, vanilla OPD, and OPD with gold-demonstration teacher context. The OPD variants are trained with a 4,096-token completion cap, and all methods are evaluated at generation caps from 4,096 to 38,912 tokens. Top row: pass@1 and pass@16, averaged over AIME24, AIME25, and HMMT25. Bottom row: mean and median response length. Vanilla OPD mildly improves pass@k while largely preserving the base model's response-length curve. Adding gold-demonstration context to the teacher shortens responses and reduces the pass@1 gains; its long-budget degradation is milder than in the OPSD setting, consistent with the use of a stronger larger teacher.

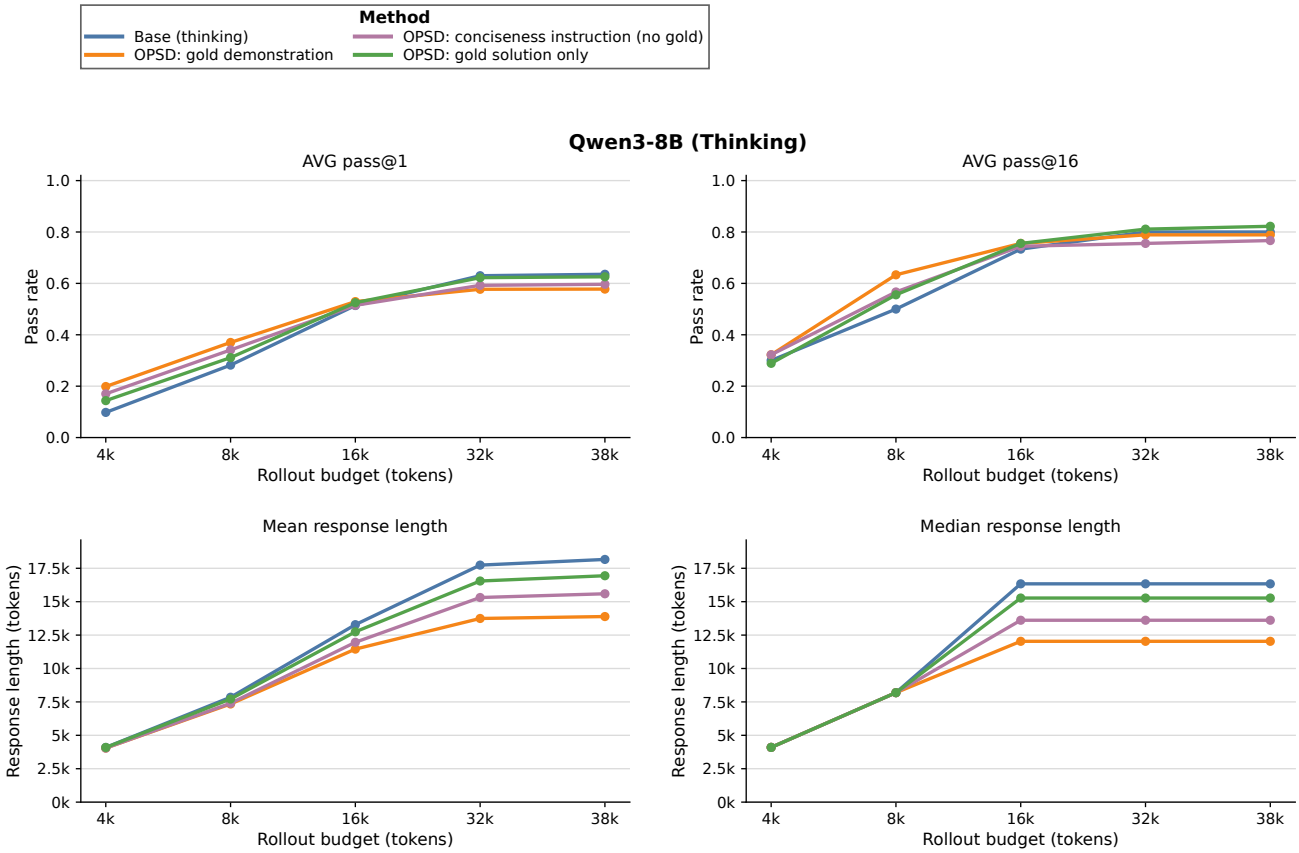

Figure 7. **A CRISP-style conciseness prompt compresses Qwen3-8B responses but does not recover the long-budget gains.** We compare base Qwen3-8B thinking, OPSD with full gold demonstrations, OPSD with final-answer-only privileged context, and a conciseness-instruction condition with no gold context, following the CRISP prompt direction of (Sang et al., 2026). Top panels report pass@1 and pass@16 averaged over AIME24, AIME25, and HMMT25; bottom panels report mean and median response length. The conciseness condition shortens 32k–38k rollouts relative to the base and gold-solution-only runs but is less compressive than full gold demonstrations. Its accuracy follows the same tradeoff: it improves short-budget performance but, at long budgets, remains below the base and gold-solution-only curves, suggesting that making the student concise alone is not enough to preserve the gains from longer thinking rollouts.

H. Sparse versus dense privileged context

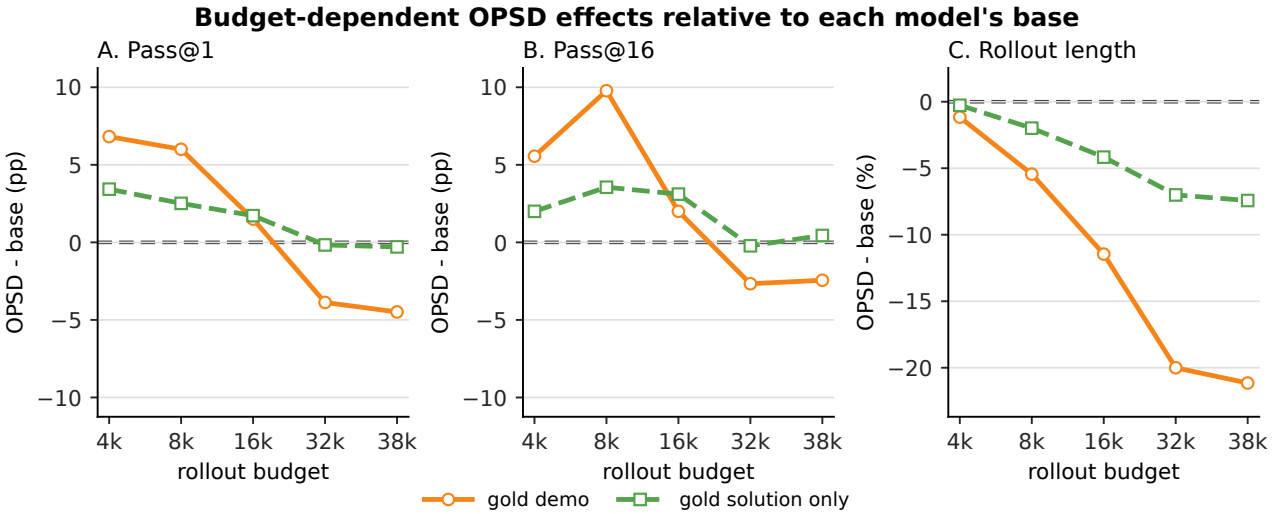

Figure 8. **In thinking models, sparse privileged context preserves long-budget behavior better than dense demonstrations.** We plot OPSD minus each model's base performance across rollout budgets, averaged over the thinking models in Table 1. Panels A and B show pass@1 and pass@16 changes in percentage points; Panel C shows relative rollout-length change. Dense gold demonstrations give larger short-budget gains but turn negative at long budgets and strongly shorten responses, while the sparse gold-solution hint remains closer to the base model.

I. Additional main-text tables

Table 21. **OPSD helps instruct models more reliably than thinking models on Countdown.** We use OPSD to train various models on Countdown and evaluate on held-out Countdown data, AIME24, AIME25, and HMMT25. Entries report avg@16 accuracy, and the Average column averages the four benchmarks. Instruct Qwen and OLMo models improve under OPSD, while the corresponding thinking models show smaller or mixed gains.

Model		Countdown	AIME24	AIME25	HMMT25	Average
Qwen3-4B-Instruct-2507	**Base**	0.736	**0.604**	0.463	**0.304**	0.527
	+OPSD	**0.865**	0.594	**0.483**	0.294	**0.559**
Qwen3-4B-Think-2507	**Base**	0.945	**0.804**	**0.804**	**0.552**	**0.776**
	+OPSD	**0.947**	0.800	0.775	0.537	0.765
OLMo-7B-Instruct	**Base**	0.719	**0.525**	**0.415**	0.237	0.474
	+OPSD	**0.814**	0.510	0.394	**0.256**	**0.494**
OLMo-7B-Think	**Base**	0.877	0.719	0.667	**0.452**	0.679
	+OPSD	**0.890**	**0.742**	**0.698**	**0.452**	**0.695**

Table 22. **The degradation depends on whether thinking is enabled during OPSD training.** We train on OpenThoughts 30k and evaluate with thinking mode in all rows. When thinking is enabled during OPSD training, Qwen3-4B loses avg@16 accuracy on all three math evaluations; disabling thinking during training preserves the base-model average while still evaluating with thinking enabled.

Method	AIME24	AIME25	HMMT25	Avg.
Qwen3-4B	0.727	0.635	0.410	0.591
+OPSD (Thinking)	0.667	0.571	0.358	0.532
+OPSD (No Thinking)	0.731	0.615	0.423	0.590

Table 23. **OPSD reduces explicit deliberation markers in paired thinking-model rollouts.** We compare base and OPSD rollouts matched by model, benchmark, problem, and sample index across the five thinking models in the OpenThoughts 15k comparison. Counts are normalized per 1,000 generated tokens. Deltas are OPSD minus base, with confidence intervals from a clustered bootstrap over model–benchmark–problem clusters. All three marker families decrease after length normalization.

Marker family	Base /1k	OPSD /1k	Δ /1k	Raw Δ
Verification	1.63	1.35	-0.28 [-0.32, -0.25]	-8.8 [-9.5, -8.1]
Backtracking	6.45	6.29	-0.16 [-0.32, -0.02]	-23.5 [-29.0, -19.1]
Hedging	3.79	3.62	-0.17 [-0.25, -0.10]	-17.0 [-18.5, -15.5]

J. Epistemic-token usage under privileged context

Table 24. **Gold-demonstration context suppresses epistemic-token usage more than vanilla OPD.** This token-level companion to Table 2 reports marker usage by Qwen3-1.7B on OpenThoughts. The aggregate epistemic-token density is the fraction of generated tokens in the epistemic-marker set. The remaining columns report occurrences per 1,000 generated tokens for representative revision and search markers such as *wait*, *recall*, *check*, and *hmm*. Vanilla OPD leaves the aggregate density essentially unchanged, while the gold-demonstration variant lowers both aggregate density and several named markers.

Method	Epistemic token density	wait	recall	okay	altern	check	verify	hmm
Base	1.080%	3.85	1.11	2.17	0.64	1.33	0.25	1.45
+OPD	1.074%	3.79	1.04	2.17	0.72	1.19	0.16	1.68
+OPD gold demo	0.850%	2.54	0.96	2.07	0.66	1.05	0.10	1.11

K. Teacher-student rollout asymmetry

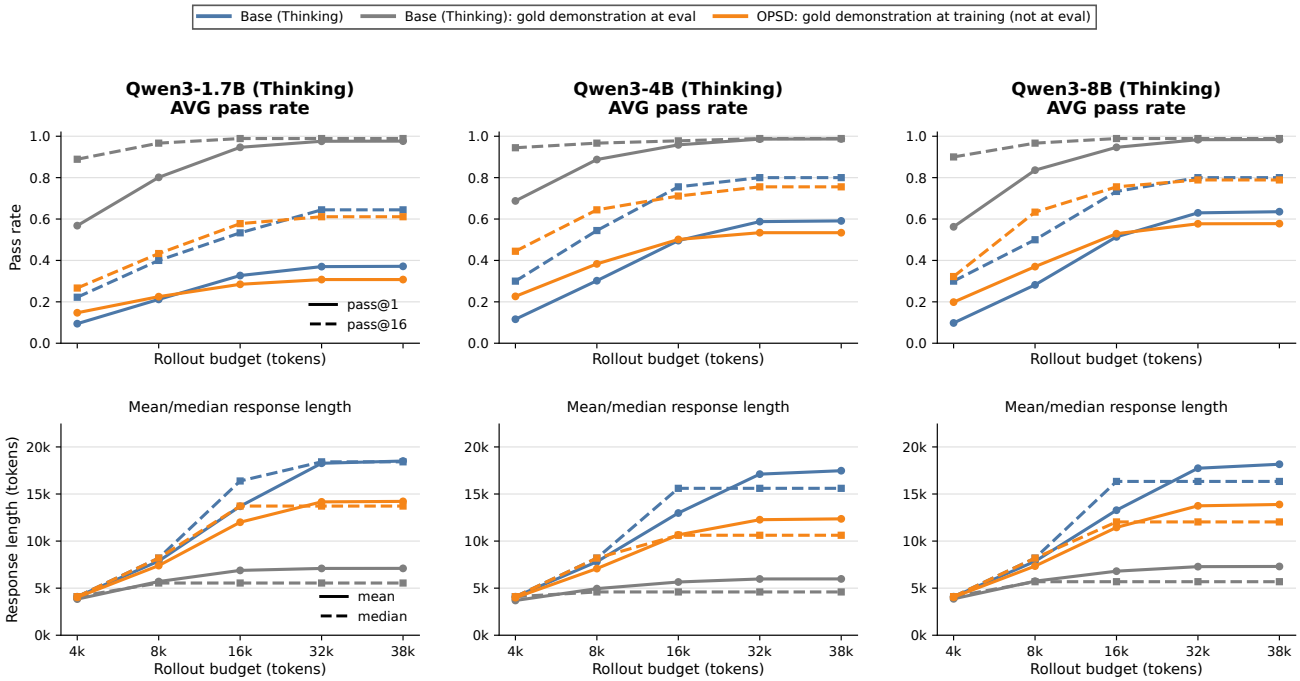

Figure 9. **OPSD students inherit the teacher's shorter response lengths, but not the pass@k benefits at longer rollout budgets.** For each Qwen3 thinking-model size (1.7B, 4B, and 8B), we compare three settings: the base model; the same base model given a gold demonstration in context, matching the setup of the OPSD teacher; and the OPSD student, evaluated without the gold demonstration in context. Top row: pass@1 and pass@16, averaged over AIME24, AIME25, and HMMT25. Bottom row: mean and median response length. Providing the teacher with the gold demonstration in context shortens the teacher's own rollouts without hurting pass@k, and can even improve pass@k. Though the OPSD student produces shorter responses like the teacher, the student's pass@k gains are smaller at short budgets and disappear or reverse at longer rollout budgets.

We next evaluate whether the gold-demo-conditioned base model—the OPSD teacher setup—exhibits the long-budget degradation observed in the OPSD student. Figure 9 compares three settings for each Qwen3 thinking-model size: the base model without privileged context, the OPSD teacher (the base model when provided the in-context gold demonstration), and the OPSD-trained student evaluated without the in-context gold demonstration. The teacher's own rollouts become much shorter, but its pass@k remains high and often improves. The student also produces shorter responses, but compared with the teacher's gains over the unprivileged base, the student's pass@k gains are smaller at short budgets and disappear or reverse at longer budgets. Thus, the student inherits the teacher's shorter response behavior more reliably than the teacher's privileged-context performance benefits.

L. Discussion

These findings suggest that the interaction between privileged supervision and test-time search is more delicate for thinking models than for instruction-tuned models. In our experiments, privileged-context OPD reduces the long-budget gains of thinking models even when closely related unprivileged OPD improves the same student. The token-level analyses provide a plausible explanation: conditioning the teacher on information unavailable to the student changes the learning signal at forking positions, high-entropy decision points where different continuations can send the rollout down different reasoning paths. Lexical markers such as "wait" or "maybe" make some of these positions visible, but the concern is broader than uncertainty language: privileged feedback may reshape which branches of the student's search are reinforced. Thus, the failure mode is not simply that privileged distillation shortens responses or suppresses epistemic markers, but that it can change the token-level structure of the reasoning process. Future self-distillation methods for strong thinking models may need to account for forking positions explicitly, so that privileged teacher feedback improves solution-directed behavior without collapsing branch-relevant test-time search.

More broadly, our results speak to OPSD-style training for long-horizon agents. In our experiments, long-rollout degradation depended on the form of privileged teacher-side context: short final-answer-only context preserved long-budget behavior better than a full gold demonstration. This distinction is relevant to emerging long-horizon agent training recipes. For example, the Composer 2.5 training stack uses targeted textual feedback, where a *short* hint is inserted into the teacher-side context and the original-context policy is distilled toward the hinted distribution (Cursor Team, 2026). Long-horizon OPSD-style training may therefore be viable, but likely requires controlling both how much privileged information the teacher receives and where the resulting token-level signal is applied.

M. Limitations

While our work is primarily centered around reporting negative results and proposing a convincing hypothesis for these failures, we do recognize limitations in our approach. First, our analysis of failures is not perfectly isolated or proved to be causal. We rely on several well-established works that investigate the importance of "forking" tokens in thinking models' reasoning abilities, and we find that OPSD methods routinely suppress this behavior. Second, we show only that existing OPSD methods do not work well for improving thinking models. We do not propose a solution to these failure cases. The question of how to leverage privileged information in distillation settings for thinking models remains open. Finally, our experiments are focused on the verifiable setting of math, where failures and successes of privileged-context self-distillation are easy to measure. We do not experiment with the success of privileged-context self-distillation on, for example, continual learning tasks, which have been proposed as a use case (albeit for non-thinking model variants) (Shenfeld et al., 2026).