The Many Faces of On-Policy Distillation: Pitfalls, Mechanisms, and Fixes

Siqi Zhu[1] **Xuyan Ye**[2*†] **Hongyu Lu**[2*†] **Weiye Shi**[3*†] **Ge Liu**[1]
[1]UIUC [2]Renmin University of China [3]Peking University

Abstract

On-policy distillation (OPD) and on-policy self-distillation (OPSD) have emerged as promising post-training methods for large language models, offering dense token-level supervision on trajectories sampled from the model's own policy. However, existing results on their effectiveness remain mixed: while OP(S)D has shown promise in system prompt and knowledge internalization, recent studies also report instability and degradation. In this work, we present a comprehensive empirical study of when OPD and OPSD work, when they fail, and why. We find that OPD on mathematical reasoning is highly sensitive to teacher choice and loss formulation, whereas OPSD fails in our tested settings due to test-time absence of instance-specific privileged information (PI). In contrast, OPSD is effective when PI represents a shared latent rule, such as a system prompt or alignment preference. We identify three failure mechanisms: (1) distribution mismatch between teacher and student caused by conditioning on student-generated prefixes, (2) optimization instability from biased TopK reverse-KL gradients, and (3) an OPSD-specific limitation where the student learns a PI-free policy that aggregates PI-conditioned teachers, which is insufficient when PI is instance-specific. We further show that stop-gradient TopK objectives, RLVR-adapted teachers, and SFT-stabilized students mitigate these failures.

1 Introduction

On-policy distillation (OPD) [1, 2] is a promising post-training algorithm for large language models (LLMs). Instead of training on off-policy responses, OPD trains the student on trajectories sampled from its own policy and uses one or more teacher models to provide dense token-level supervision along these on-policy rollouts. This makes OPD appealing as a practical mechanism for integrating the capabilities of multiple stronger teachers into the student [3, 4], mitigating catastrophic forgetting, and improving sample efficiency [5, 6]. A closely related topic is on-policy self-distillation (OPSD), where the teacher is not a stronger model but the student model itself conditioned on additional privileged information (PI) [7–9]. The PI may include ground-truth answers, system prompts or user preferences. These methods are attractive for post-training because they can convert teacher behavior or training-time context into supervision on the student's own state distribution.

Despite the conceptual simplicity, the empirical behavior of OPD and OPSD remains poorly understood. Prior work has reported successful applications of self-distillation for context internalization and reasoning [7–9]. However, recent studies argue that on-policy (self) distillation can be unstable and may degrade performance [10, 11]. These mixed results raise a basic question: **when do OPD and OPSD actually work, when do they fail, and what mechanisms explain the difference**?

*Co-second authorship
†Work done during an internship at UIUC.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

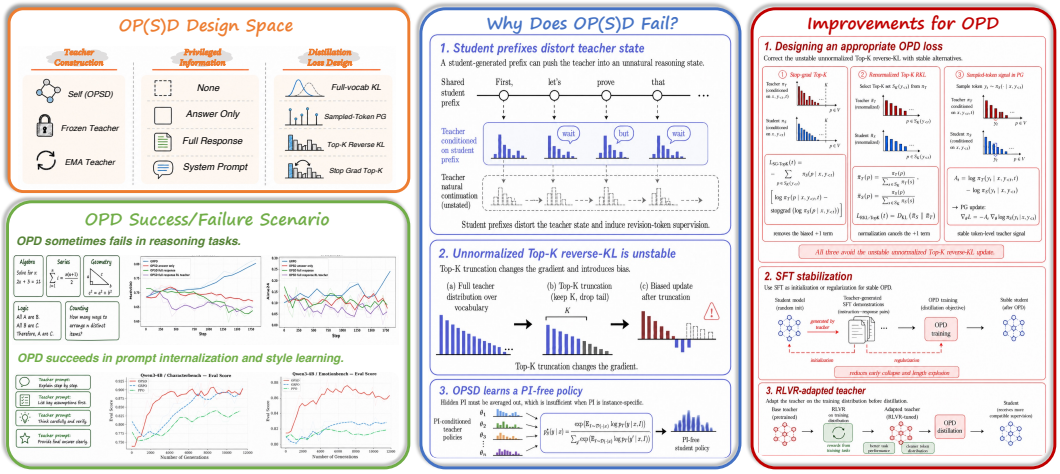


Figure 1: **Overview.** We map the OP(S)D *design space* (left, top) and its task-dependent success/failure behavior (left, bottom), identify three failure mechanisms—prefix-distorted teacher state, biased Top-K reverse-KL, and PI-marginalized OPSD policy (middle), and propose practical fixes: stable Top-K losses, SFT stabilization, and RLVR-adapted teachers (right).

In this paper, we present a comprehensive empirical study of OPD and OPSD across both successful and failing regimes. On reasoning tasks, OPD provides only conditional benefits and can suffer from length explosion, repetition, and teacher-student mismatch, while OPSD fails in our tested math reasoning settings. In contrast, OPSD is effective for system-prompt internalization and alignment, where the privileged information induces a shared latent behavior that can be compressed into a PI-free policy at test time.

We identify three mechanisms behind these outcomes. First, student-generated prefixes can move the teacher into states where its token-level supervision is locally incompatible with the student's trajectory. Second, Top-K reverse-KL approximations can introduce biased gradients and destabilize optimization. Third, OPSD only learns a single PI-free consensus policy across PI-conditioned teachers, which is insufficient when PI is instance-specific.

Motivated by these findings, we study practical stabilizers for OPD: a stop-gradient Top-K KL surrogate that removes the biased gradient term, Reinforcement Learning with Verifiable Rewards (RLVR) adaptation of the teacher to improve task performance, and Supervised Fine-Tuning (SFT) the student to improve output format and stabilize response-length dynamics.

Our contributions are:

- We empirically study when OPD and OPSD succeed or fail in reasoning, system-prompt internalization, and alignment.
- We identify three failure mechanisms: prefix-induced teacher-student mismatch, biased TopK reverse-KL gradients, and OPSD's PI-free aggregation across instance-specific PI.
- We propose practical stabilizers based on stop-gradient Top-K KL, RLVR-adapted teachers, and student SFT for output well-formedness and response length stabilization.

2 Related Work

On-Policy Distillation and Context Distillation. **On-policy distillation** trains the student on its own sampled trajectories and uses a teacher to provide dense supervision. Agarwal et al. [1] show that this formulation can outperform standard off-policy distillation, and later work demonstrates its effectiveness for math reasoning [5]. A closely related topic is **context distillation**, which turns in-context behaviors into model parameters. Prior work shows that models can distill instructions and knowledge from context into persistent capabilities [12]. Recent work extends this idea to on-policy distillation for continual learning [6, 9].

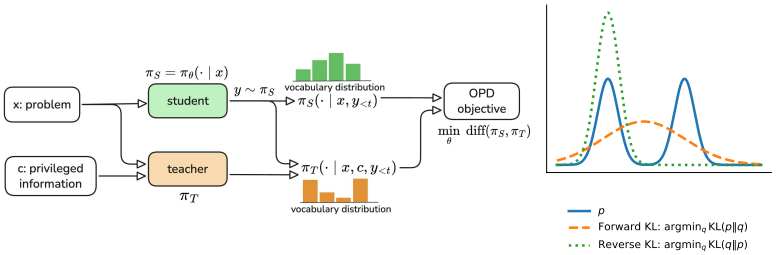


Figure 2: (**Left**) On-Policy (Self-)Distillation. In OPSD, the teacher is constructed from the student itself and privileged information (PI) is necessary. In OPD, the teacher is a stronger model and PI is optional. (**Right**) **p**: teacher distribution, **q**: student distribution. Reverse KL is mode-seeking, whereas forward KL is mode-covering.

Reinforcement Learning from Textual Feedback. Another related direction augments reinforcement learning with language-based feedback instead of relying only on scalar rewards. Song et al. [13] show that textual feedback can provide richer supervision over intermediate behaviors and improve agent capabilities. In reasoning tasks, POPE [14] uses privileged guidance to improve exploration on hard problems. Compared with these methods, our work also studies the role of additional textual information, but focuses on on-policy distillation rather than reinforcement learning objectives.

3 Preliminary: On-Policy Distillation

Let $x \sim \mathcal{D}$ be an input, π_θ be the student policy, and $\pi_T(\cdot \mid x, I)$ be the teacher policy with optional privileged information I. In OPD, π_T is a stronger external model. In OPSD, π_T is a student-derived policy augmented with PI. As illustrated in Figure 2, OP(S)D trains on student-generated trajectories $y = (y_1, \ldots, y_T) \sim \pi_\theta(\cdot \mid x)$. At each prefix $y_{<t}$, the student receives token-level supervision from the teacher:

$$\mathcal{L}_{\text{OP(S)D}} = \mathbb{E}_{x,y}\left[\frac{1}{T}\sum_{t=1}^{T} \ell_t(\pi_\theta(\cdot \mid x, y_{<t}), \text{stograd}(\pi_T(\cdot \mid x, y_{<t}, I)))\right]. \tag{1}$$

Full-vocabulary KL. A standard choice for ℓ_t is to match the teacher and student vocabulary with reverse KL:

$$D_{\text{KL}}(\pi_\theta \| \pi_T) = \sum_v \pi_\theta(v) \log \frac{\pi_\theta(v)}{\pi_T(v)}. \tag{2}$$

Forward KL is mode-covering and encourages the student to cover the teacher distribution. Reverse KL is mode-seeking and tends to preserve the student's high-probability modes, making it less prone to catastrophic forgetting [15]. In OPD, forward KL is undesirable because it pushes the student toward teacher-preferred but student-unlikely tokens.

For reverse KL, writing $p_\theta(v) = \pi_\theta(v \mid x, y_{<t})$ and $p_T(v) = \pi_T(v \mid x, y_{<t}, I)$, its gradient is

$$\nabla_\theta D_{\text{KL}}(p_\theta \| p_T) = \sum_v p_\theta(v) \left[\log \frac{p_\theta(v)}{p_T(v)} \cancel{+1}\right] \nabla_\theta \log p_\theta(v). \tag{3}$$

In the full vocabulary, the constant **+1** has zero contribution since $\sum_v p_\theta(v) \nabla_\theta \log p_\theta(v) = 0$.

Sampled-token KL. Another choice of ℓ_t uses the sampled token $y_t \sim \pi_\theta$. The sampled-token objective treats the teacher-student log-probability gap as an advantage:

$$\mathcal{L}_{\text{PG}} = -\mathbb{E}_{y\sim\pi_\theta} \sum_t \textbf{stopgrad}\big(\log \pi_T(y_t \mid x, y_{<t}, I) - \log \pi_\theta(y_t \mid x, y_{<t})\big) \log \pi_\theta(y_t \mid x, y_{<t}), \tag{4}$$

whose policy-gradient form is

$$\nabla_\theta \mathcal{L}_{\text{PG}} = -\mathbb{E}_{y\sim\pi_\theta} \sum_t \textbf{stopgrad}\big(\log \pi_T(y_t \mid x, y_{<t}, I) - \log \pi_\theta(y_t \mid x, y_{<t})\big) \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}). \tag{5}$$

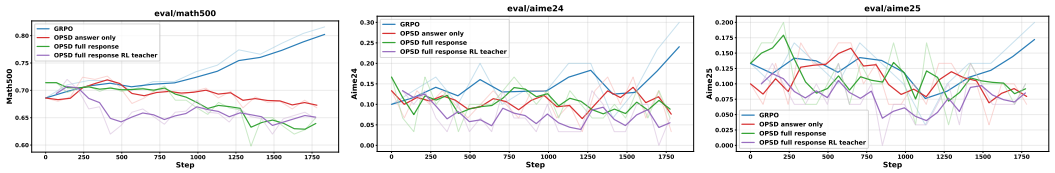
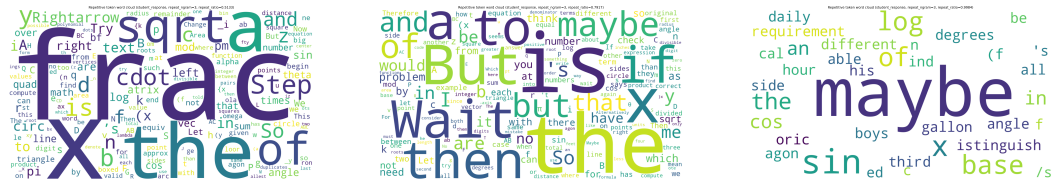
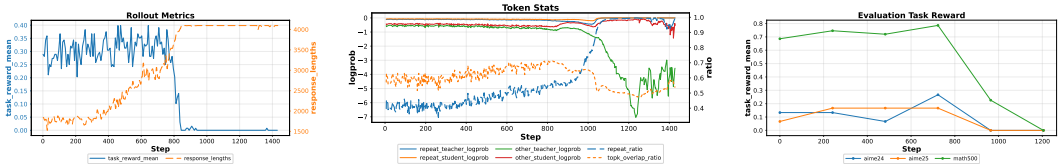


Figure 3: Qwen3-1.7B, trained on OpenThoughts. OPSD fails to improve student.



(a) step0 (b) step700 (c) step1000



(d) rollout metrics (e) token stats (f) eval acc

Figure 4: Collapse under unnormalized Top-20 reverse KL. The model first becomes verbose, then degenerates into repetitive "maybe" outputs as response length reaches the limit and evaluation accuracy drops. Token statistics show that repetitive tokens dominate as the repeat ratio approaches one.

4 Experiments

We evaluate OPD and OPSD on reasoning, system-prompt internalization, and alignment, covering both failure and success regimes.

4.1 Math Reasoning

OPSD. We train Qwen3-1.7B on OpenThoughts using stopgrad Top20 reverse KL (Equation 12), with the step-0 checkpoint as the PI-conditioned teacher. We keep English problems with verifiable boxed answers. As shown in Figure 3, neither answer-only nor full-response PI makes stable improvements on Math500, AIME24, or AIME25. Full-response PI performs worse than answer-only PI, suggesting that richer PI can increase mismatch rather than improve supervision. We further test whether this failure is due to a weak PI-conditioned teacher by training a Qwen3-1.7B teacher with GRPO under full-response PI (purple line in Figure 3). This RL-trained teacher performs even worse, indicating that PI alone does not make math OPSD effective.

OPD. We use a Qwen3-1.7B student and a Qwen3-8B teacher, both with thinking mode disabled, trained on OpenThoughts with an unnormalized Top20 reverse-KL objective (Figure 4). OPD initially improves but later collapses. Around step 700, rollout length and revision tokens such as "wait" and "maybe" increase sharply; by step 1000, the model degenerates into repetitive "maybe" outputs and accuracy on Math500, AIME24, and AIME25 drops nearly to zero. Token statistics show a corresponding abrupt rise in repetition and a collapse of teacher-student overlap.

4.2 Alignment & System Prompt Internalization.

Alignment. We evaluate OPSD on two style-alignment benchmarks: CharacterBench [16] and EmotionBench [17]. For both tasks, we provide the teacher with a question-specific alignment prompt as privileged information; details are provided in Appendix A.20. We compare OPSD with GRPO

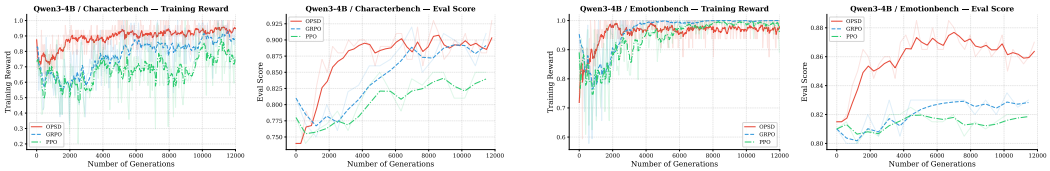


Figure 5: Training reward (left) and evaluation score (right) curves for OPSD, GRPO, and PPO on CharacterBench and EmotionBench, using Qwen3-4B-Instruct as student models.

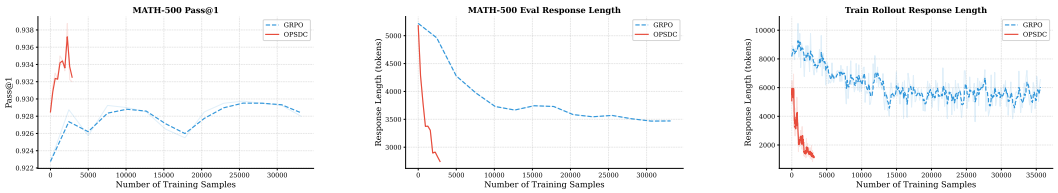


Figure 6: Comparison of GRPO and OPSD on Qwen3-8B (thinking mode) trained with DAPO-Math-17k [20]. From left to right: MATH-500 Pass@1, evaluation response length, and training rollout response length.

and PPO using Qwen3-4B-Instruct and Qwen3-8B students. As shown in Figure 5, under the same sampling budget, OPSD improves and converges faster than both RL baselines.

System Prompt Internalization. We next study two applications of system prompt internalization: reasoning compression and safety alignment on adversarial questions. In contrast to the alignment tasks above, where the privileged information is question-specific, system prompt internalization uses a fixed system prompt as privileged information for all questions. For reasoning compression, shown in Figure 6, OPSD does not harm accuracy while substantially reducing response length. Compared with GRPO using a length penalty, OPSD is more sample efficient. It also offers a practical efficiency advantage: RL requires a large generation length to get a final answer, but OP(S)D can use shorter rollouts because the supervision is provided token-wise. However, we find that this application requires a sufficiently capable model, such as an 8B-scale model, to be effective. This finding is also consistent with [18]. The second application is safety alignment. We conduct experiments with Qwen3-1.7B and Wildguardmix dataset [19] using the system prompt in Appendix A.18, with results shown in Figure 7. OPSD improves rapidly in the early stage of training, but its final performance is constrained by the teacher. In contrast, GRPO improves more gradually but continues to make gains after OPSD saturates.

5 Mechanism of On-Policy Distillation

5.1 Why Does OPD Sometimes Fail?

Student Prefixes Distort the Teacher State. A central challenge in OPD is that the teacher is conditioned on student-generated prefixes rather than its own trajectory. Although a strong teacher may solve the problem independently, a partial student trajectory can force it into an intermediate reasoning state that it would not have reached on its own. To quantify this effect, we conduct a prefix-conditioned teacher evaluation on GPQA-Diamond, using Qwen3-1.7B as the student and Qwen3-14B as the teacher. The standalone teacher achieves **62.1%** accuracy, whereas the student-prefix-conditioned teacher drops to **46.0%** accuracy (details in Appendix A.22). Thus, OPD supervision can become weaker than the teacher's standalone capability if the teacher continues from student prefixes. Figure 8 illustrates this failure mode at the token level. Given a student prefix that has already committed to one reasoning branch, the teacher assigns high probability to tokens such as "wait" and "but," which revise or redirect the trajectory rather than extend it. This creates a local semantic conflict between the student's current path and the teacher's preferred path. In math reasoning, such unreliable supervision encourages verbose and inconsistent reasoning.

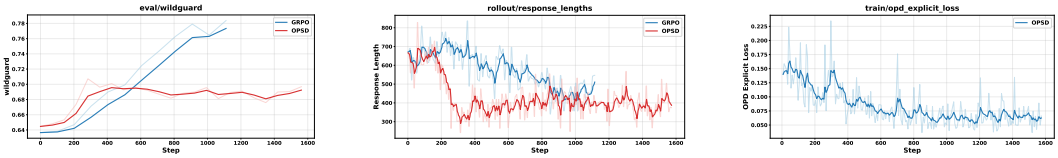


Figure 7: Train and evaluate Qwen3-1.7B (nothink) on Wildguardmix using their original train and test splits.

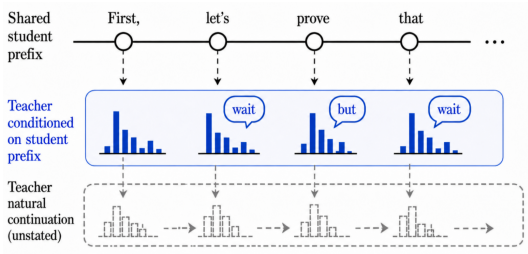


Figure 8: Local semantic conflict in OPD. If the teacher's preferred path is inconsistent with the student prefix, teacher may encourage branch switching rather than branch refinement. This often appears as high probability assigned to revision tokens such as **"wait"** and **"but"**, which attempt to redirect the trajectory.

The pitfall of unnormalized TopK Reverse KL. To reduce the GPU memory peak of reverse-KL distillation, we usually use the teacher's or the student's TopK set to compute a surrogate OPD loss. However, this may change the optimization stability and lead to model collapse in Figure 4. Consider the full-vocabulary reverse KL in Equation 2. A TopK truncation keeps only a subset $S_K(y_{<t}) \subset \mathcal{V}$:

$$\mathcal{L}_{\text{Top-}K\text{-RKL}}(t) = \sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \log \frac{\pi_S(v \mid x, y_{<t})}{\pi_T(v \mid x, y_{<t}, I)}. \tag{6}$$

The difference is in the gradient. For the full-vocabulary objective in Equation 2, the constant term **+1** can be removed because

$$\sum_v p_\theta(v) \nabla_\theta \log p_\theta(v) = 0 \tag{7}$$

However, this no longer holds for TopK:

$$\sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \nabla_\theta \log \pi_S(v \mid x, y_{<t}) = \nabla_\theta \sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \neq 0. \tag{8}$$

The **+1** term in the gradient remains:

$$\nabla_\theta \mathcal{L}_{\text{Top-}K\text{-RKL}}(t) = \sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \left(\log \frac{\pi_S(v \mid x, y_{<t})}{\pi_T(v \mid x, y_{<t}, I)} \textbf{+1} \right) \nabla_\theta \log \pi_S(v \mid x, y_{<t}), \tag{9}$$

Thus TopK truncation changes the update rule: a token is promoted only if $\pi_T(v \mid x, y_{<t}, I) > e\, \pi_S(v \mid x, y_{<t})$. Teacher-preferred tokens with smaller teacher-student margins are still suppressed, biasing the student away from the teacher distribution and potentially toward unstable low-probability continuations.

5.2 OPSD Learns a PI-Free Policy.

Let x denote the question and I denote the privileged information (PI). Since the student does not observe I, OPSD optimizes

$$\min_{p_S} \mathbb{E}_{(x,I)\sim\mathcal{D}} \Big[D_{\text{KL}}\big(p_S(\cdot \mid x) \,\|\, p_T(\cdot \mid x, I)\big) \Big]. \tag{10}$$

The optimal student is the normalized geometric mean of the PI-conditioned teacher distributions:

$$p_S^\star(y \mid x) = \frac{\exp\left(\mathbb{E}_{I\sim\mathcal{D}(\cdot|x)} \log p_T(y \mid x, I)\right)}{\sum_{y'} \exp\left(\mathbb{E}_{I\sim\mathcal{D}(\cdot|x)} \log p_T(y' \mid x, I)\right)}. \tag{11}$$

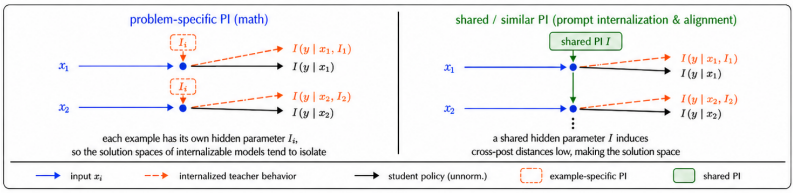


Figure 9: Effectiveness of OPSD depends on the structure of privileged information I.

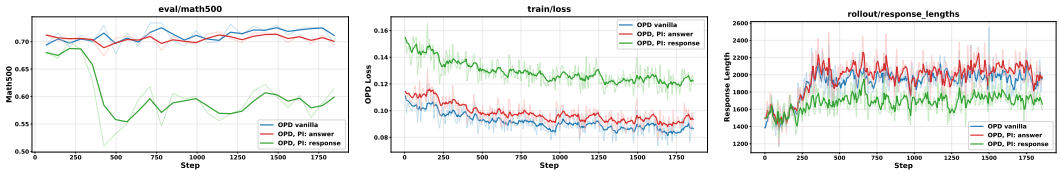


Figure 10: **PI does not improve OPD on math reasoning with a stronger teacher.** Using a Qwen3-8B teacher and a Qwen3-1.7B student on OpenThoughts, both final-answer PI and full-response PI underperform vanilla OPD. PI-conditioned OPD leads to higher KL loss.

This form indicates that OPSD can distill behavior that is consistently supported under different PI. Outputs that receive high probability under some PI but low probability under others are suppressed. Figure 9 illustrates why the structure of PI (I) matters. When PI is problem-specific, as in math reasoning, different examples depend on different PI I_1, I_2, I_3. These parameters can induce incompatible teacher behaviors, so the student is driven toward the common pattern of these incompatible teacher policies. This makes the student substantially weaker than the PI-conditioned teacher. In contrast, when PI follows a similar latent structure, as in prompt internalization or alignment, many examples are governed by the same underlying rule I. The student can then compress the teacher's PI-conditioned behavior into a reusable inductive bias.

To test whether math PI provides useful supervision, and to rule out limited teacher capability as the cause of OPSD failure, we use a stronger Qwen3-8B teacher to train a Qwen3-1.7B student with vanilla OPD, answer-PI OPD, and response-PI OPD. As shown in Figure 10, PI remains **unhelpful**: both PI-conditioned methods underperform vanilla OPD, with response-PI performing worst. Their distillation losses (at step 0) follow *OPD, PI: response* > *OPD, PI: answer* > *OPD vanilla*, suggesting that PI increases teacher-student mismatch. The higher loss of response-PI indicates that a longer PI induces a larger distributional shift rather than a better distillation signal.

6 Solutions

6.1 Designing an appropriate OPD loss.

To mitigate the pathology of TopK reverse KL, we replace exact TopK reverse-KL with a stop-gradient version. Specifically, instead of differentiating through both π_S and $\log \pi_S$, we stop gradients of the log-probability term. The resulting objective is

$$\mathcal{L}_{\text{SG-Top}K}(t) = -\sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \left[\log \pi_T(v \mid x, y_{<t}, I) - \textbf{stopgrad}\big(\log \pi_S(v \mid x, y_{<t})\big)\right], \tag{12}$$

with gradient $\nabla_\theta \mathcal{L}_{\text{SG-Top}K}(t)$:

$$-\sum_{v \in S_K(y_{<t})} \pi_S(v \mid x, y_{<t}) \left[\log \pi_T(v \mid x, y_{<t}, I) - \log \pi_S(v \mid x, y_{<t})\right] \nabla_\theta \log \pi_S(v \mid x, y_{<t}). \tag{13}$$

Although this objective is not the exact gradient of reverse KL, it is better aligned with the policy-gradient view of on-policy distillation, where the weighting term should act as an advantage rather than a differentiable part of the loss.

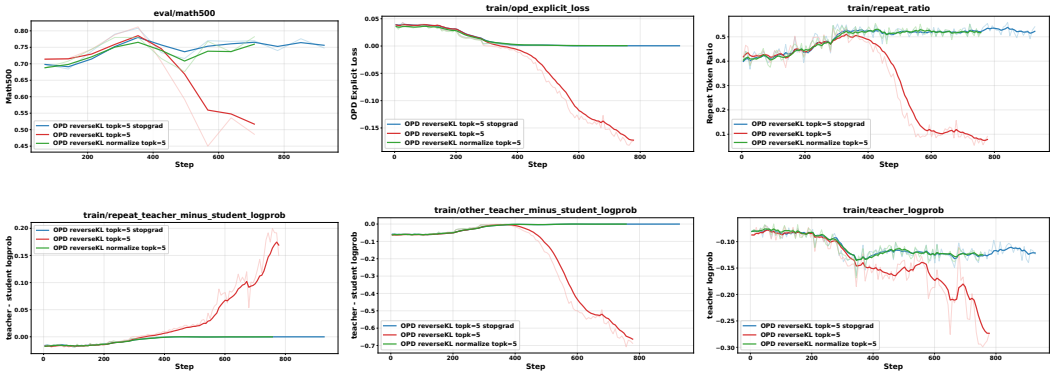


Figure 11: Teacher: Qwen3-1.7B-GRPO (nothink), Student: Qwen3-1.7B (nothink), DAPO, TopK=5.

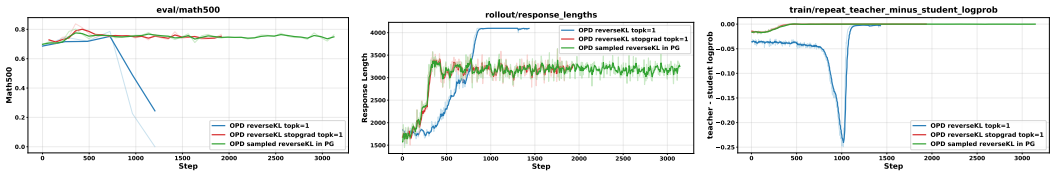


Figure 12: Whether to put the distillation loss in policy gradient? sampled token KL in policy gradient is stable, while putting sampled token KL as a single loss (TopK=1) leads to model collapse. Teacher: Qwen3-1.7B-GRPO, Student: Qwen3-1.7B, dataset: DAPO

By contrast, this issue is much less severe for two alternative formulations. The first is the TopK reverse KL with renormalization. Let

$$\bar{\pi}_S = \frac{\pi_S(v \mid x, y{<}\, t)}{\sum_{u \in S_K(y<t)} \pi_S(u \mid x, y{<}\, t)}, \bar{\pi}_T = \frac{\pi_T(v \mid x, y{<}\, t, I)}{\sum_{u \in S_K(y<t)} \pi_T(u \mid x, y{<}\, t, I)}, v \in S_K(x, y{<}\, t). \tag{14}$$

The renormalized TopK reverse KL is

$$\mathcal{L}_{\text{Renorm-Top-}K\text{-RKL}}(t) = \sum_{v \in S_K(y<t)} \bar{\pi}_S(v \mid x, y{<}\, t) \log \frac{\bar{\pi}_S(v \mid x, y{<}\, t)}{\bar{\pi}_T(v \mid x, y{<}\, t, I)}. \tag{15}$$

Since distributions are normalized within the TopK set, the constant $+1$ term cancels:

$$\sum_{v \in S_K(y_{<t})} \bar{\pi}_S(v \mid x, y_{<t}) \nabla_\theta \log \bar{\pi}_S(v \mid x, y_{<t}) = \nabla_\theta \sum_{v \in S_K(y_{<t})} \bar{\pi}_S(v \mid x, y_{<t}) = 0. \tag{16}$$

The gradient does not contain the $+1$ that causes the unnormalized TopK reverse-KL bias in paragraph 5.1. However, it only matches the relative probabilities inside the selected TopK set and ignores the probability mass assigned to that set. As a result, it mitigates the local gradient bias but no longer faithfully approximates the full-vocabulary reverse KL.

A second alternative is to move the sampled-token signal into the policy-gradient (Equation4). This formulation also avoids the TopK reverse-KL issue. The drawback is that this objective uses only the sampled token y_t. It therefore captures sparse teacher signal rather than the teacher vocabulary distribution.

Another important design choice is the TopK truncation. Ideally, we would like to query the teacher on the student's per-position TopK set $S_{\text{stu},K}(y_{<t})$, or vice versa. However, this is not directly supported when using SGLang as inference engine (details in appendix A.6).

Instead, we must take the union of all per-position TopK sets $U = \bigcup_{t=1}^{T} S_{\text{stu},K}(y_{<t})$, and query this same global set at every position. If the response length is T, the ideal cost is $T \times K$, but the actual queried tensor becomes $T \times |U|$. In the worst case, the TopK sets at different positions are disjoint,

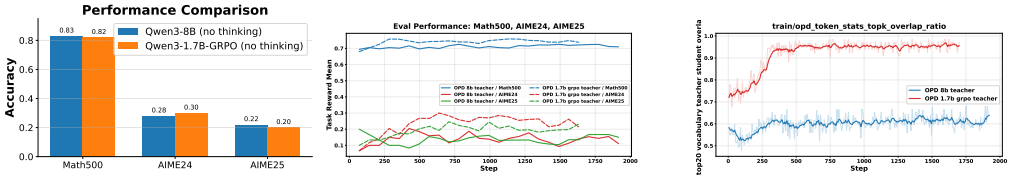


Figure 13: dataset: OpenThoughts. **Left**: Qwen3-8B and Qwen3-1.7B-GRPO have similar math reasoning performance. **Middle**: In OPD, Qwen3-1.7B-GRPO is a more effective teacher. **Right**: Qwen3-1.7B-GRPO's Top20 vocabulary distribution is more aligned with the Qwen3-1.7B student.

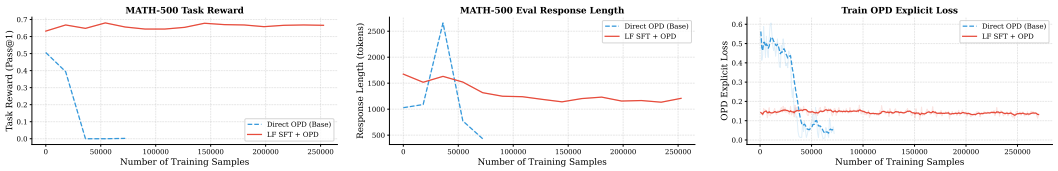


Figure 14: Qwen3-4B teacher, Qwen3-1.7B-Base student, OpenThoughts. **Direct OPD** uses the student without SFT warm-up; **SFT + OPD** fine-tunes on teacher-generated traces before OPD.

so $|U| = \min(|\mathcal{V}|, TK)$, which increases memory by a factor of $\frac{\min(|\mathcal{V}|,TK)}{K}$. As a simple fix, we backpropagate only through tokens that appear in both models' TopK sets:

$$S_K(y_{<t}) = S_{\text{tea},K}(y_{<t}) \cap S_{\text{stu},K}(y_{<t}) \tag{17}$$

Interestingly, prior work [21] finds that the teacher-student TopK intersection performs comparably to the student TopK, suggesting that our design does not degrade training effectiveness.

We compare three TopK objectives with $K = 5$ in Figure 11: unnormalized reverse KL (Equation 6), its stop-gradient version (Equation 12), and its renormalized version (Equation 15). The unnormalized objective collapses during training, whereas both modifications achieve comparable stable performance. We further test the formulation that places the distillation signal within the policy gradient (Equation 4). As shown in Figure 12, it performs similarly to Top1 reverse KL with stop-gradient, while the unmodified Top1 reverse KL still collapses. These results suggest that correcting the biased reverse-KL gradient, through stop-gradient, renormalization, or a policy-gradient formulation, is essential for stable distillation.

6.2 Enhance teacher with RLVR.

One effective way to improve OPD is to adapt the teacher on the training distribution before distillation. RLVR improves the teacher's task performance on the training set and can make the teacher's output distribution closer to that of the student. This reduces the teacher-student distribution mismatch and makes token-level distillation signals more locally compatible with the student's on-policy prefixes.

We evaluate this strategy in Figure13. We first train a Qwen3-1.7B model with DAPO [20] for 200 steps, obtaining a RL-adapted teacher denoted Qwen3-1.7B-GRPO. Qwen3-1.7B-GRPO achieves performance on Math500, AIME24, and AIME25 that is comparable to Qwen3-8B. We then use either Qwen3-1.7B-GRPO or Qwen3-8B as the teacher to distill a Qwen3-1.7B student on the OpenThoughts [22] dataset.

Although the two teachers have similar reasoning performance, distillation from Qwen3-1.7B-GRPO significantly outperforms distillation from Qwen3-8B, suggesting that teacher accuracy alone is not sufficient to predict OPD effectiveness. A teacher closer to the student distribution can provide stronger supervision, even if it does not have stronger benchmark accuracy. Compared with using PI during distillation (Figures 3 and 10), this pipeline provides a stronger teacher and serves as an effective alternative to PI-based OP(S)D.

6.3 Stabilize Student with SFT.

Prior work has used supervised data to stabilize OPD. For example, Luo et al. [23] combine OPD with SFT on gold answers to mitigate length inflation, while Li et al. [21] fine-tune the student on teacher-generated traces to reduce the teacher-student distribution mismatch. In the Qwen3-4B $\rightarrow$ Qwen3-1.7B-Base setting, we observe a distinct instability source: the 1.7B student initially produces garbled, nonsensical outputs (non-English Unicode) in some questions. Unlike ordinary incorrect reasoning, these outputs contain degenerate sequences with little semantic structure. In this case, teacher is unlikely to provide a meaningful signal, making Direct OPD unstable. As shown in Figure 14, teacher-trace SFT regularizes the student's output space, stabilizes response length, and allows subsequent OPD to improve accuracy. These results suggest that SFT helps OPD by keeping on-policy samples in well-formed regions where teacher feedback remains effective. Experiment details are in Appendix A.23.

7 Conclusion

We present an empirical study of OPD and OPSD for LLMs, showing that the effectiveness depends on task structure and the nature of privileged information. OPSD is ineffective in our tested mathematical reasoning settings, where privileged information is largely instance-specific, but works well for system prompt internalization and alignment tasks where privileged information reflects shared latent rules. We identify three failure mechanisms: teacher-student distribution mismatch, biased TopK reverse-KL gradients, and the limitation of OPSD in learning only the marginal distribution over instance-specific privileged information. We further show that stop-gradient TopK surrogate loss, RLVR teacher improvement, and SFT stabilization can improve practical training stability. Our findings are based on a limited set of model families and model scales, and larger-scale settings may exhibit different behavior. Future work may explore iterative self-improvement pipelines that combine SFT, RL, and OPD: SFT provides a stable initialization, RL improves task-specific behavior, and OPD distills the improved on-policy behavior back into the student.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024.

[2] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models, 2026.

[3] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

[4] Core Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, Gang Xie, Hailin Zhang, Hanglong Lv, Hanyu Li, Heyu Chen, Hongshen Xu, Houbin Zhang, Huaqiu Liu, Jiangshan Duo, Jianyu Wei, Jiebao Xiao, Jinhao Dong, Jun Shi, Junhao Hu, Kainan Bao, Kang Zhou, Lei Li, Liang Zhao, Linghao Zhang, Peidian Li, Qianli Chen, Shaohui Liu, Shihua Yu, Shijie Cao, Shimao Chen, Shouqiu Yu, Shuo Liu, Tianling Zhou, Weijiang Su, Weikun Wang, Wenhan Ma, Xiangwei Deng, Bohan Mao, Bowen Ye, Can Cai, Chenghua Wang, Chengxuan Zhu, Chong Ma, Chun Chen, Chunan Li, Dawei Zhu, Deshan Xiao, Dong Zhang, Duo Zhang, Fangyue Liu, Feiyu Yang, Fengyuan Shi, Guoan Wang, Hao Tian, Hao Wu, Heng Qu, Hongfei Yi, Hongxu An, Hongyi Guan, Xing Zhang, Yifan Song, Yihan Yan, Yihao Zhao, Yingchun Lai, Yizhao Gao, Yu Cheng, Yuanyuan Tian, Yudong Wang, Zhen Tang, Zhengju Tang, Zhengtao Wen, Zhichao Song, Zhixian Zheng, Zihan Jiang, Jian Wen, Jiarui Sun, Jiawei Li, Jinlong Xue, Jun Xia, Kai Fang, Menghang Zhu, Nuo Chen, Qian Tu, Qihao Zhang, Qiying Wang, Rang Li, Rui Ma, Shaolei Zhang, Shengfan Wang, Shicheng Li, Shuhao Gu, Shuhuai Ren, Sirui Deng, Tao Guo, Tianyang Lu, Weiji Zhuang, Weikang Zhang, Weimin Xiong, Wenshan Huang, Wenyu Yang, Xin Zhang, Xing Yong, Xu Wang, Xueyang Xie, Yilin Jiang, Yixin Yang, Yongzhe He, Yu Tu, Yuanliang Dong, Yuchen Liu, Yue Ma, Yue Yu, Yuxing Xiang, Zhaojun Huang, Zhenru Lin, Zhipeng Xu, Zhiyang Chen, Zhonghua Deng, Zihan Zhang, and Zihao Yue. Mimo-v2-flash technical report, 2026.

[5] Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. https://thinkingmachines.ai/blog/on-policy-distillation.

[6] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026.

[7] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026.

[8] Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. On-policy self-distillation for reasoning compression, 2026.

[9] Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026.

[10] Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes, 2026.

[11] 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?, 2026.

[12] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022.

[13] Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback, 2026.

[14] Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration, 2026.

[15] Alan Chan, Hugo Silva, Sungsu Lim, Tadashi Kozuno, A. Rupam Mahmood, and Martha White. Greedification operators for policy optimization: Investigating forward and reverse kl divergences, 2022.

[16] Jinfeng Zhou, Yongkang Huang, Bosi Wen, Guanqun Bi, Yuxuan Chen, Pei Ke, Zhuang Chen, Xiyao Xiao, Libiao Peng, Kuntian Tang, Rongsheng Zhang, Le Zhang, Tangjie Lv, Zhipeng Hu, Hongning Wang, and Minlie Huang. Characterbench: Benchmarking character customization of large language models, 2024.

[17] Jen tse Huang, Man Ho Lam, Eric John Li, Shujie Ren, Wenxuan Wang, Wenxiang Jiao, Zhaopeng Tu, and Michael R. Lyu. Emotionally numb or empathetic? evaluating how llms feel using emotionbench, 2024.

[18] Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation, 2026.

[19] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms, 2024.

[20] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

[21] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe, 2026.

[22] 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.

[23] Feng Luo, Yu-Neng Chuang, Guanchu Wang, Zicheng Xu, Xiaotian Han, Tianyi Zhang, and Vladimir Braverman. Demystifying opd: Length inflation and stabilization strategies for large language models. *arXiv preprint arXiv:2604.08527*, 2026.

[24] 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, 2026.

[25] Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking, 2026.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[27] Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. Persuasion for good: Towards a personalized persuasive dialogue system for social good. *arXiv preprint arXiv:1906.06725*, 2019.

[28] Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models, 2026.

A Appendix

A.1 Experiment Setup

We use a maximum response length of 16384, temperature 1.0, and top-p 0.95 for evaluation.

Table 1 summarizes the main hyperparameters for OPD, OPSD, GRPO, and PPO. Unless otherwise specified, OPD and OPSD use one rollout per prompt and optimize a stop-gradient, renormalized Top-K reverse-KL objective. OPSD disables thinking mode and uses the step-0 student checkpoint as the fixed teacher model by default. All experiments are conducted on a server with 10 * NVIDIA RTX PRO 6000 Blackwell GPUs.

Category	**OPD**	**OPSD**	**GRPO**	**PPO**
Rollout batch size	64	64	32	64
Prompts per batch	64	64	32	64
Samples per prompt	1	1	8	1
Maximum rollout response length	4096	4096	8192	8192
Rollout temperature	1.0	1.0	1.0	1.0
Rollout top-p	0.95	0.95	0.95	0.95
Optimizer	Adam	Adam	Adam	Adam
Learning rate	2×10^{-6}	2×10^{-6}	2×10^{-6}	2×10^{-6}
Learning-rate schedule	cosine decay	cosine decay	cosine decay	cosine decay
Warmup fraction	0.1	0.1	0.1	0.1
Weight decay	0.1	0.1	0.1	0.1
Adam β_1 / β_2	0.9 / 0.98	0.9 / 0.98	0.9 / 0.98	0.9 / 0.98
Training objective	Top-K reverse KL	Top-K reverse KL	policy gradient	PPO
KL / regularization coefficient	–	–	0.0	0.0
Top-K	20	20	–	–
Advantage estimator	–	–	GRPO	GAE
GAE γ / λ	–	–	–	1.0 / 0.95
Teacher privileged information	none	final answer / full response	–	–
Teacher model	external teacher	step-0 student	–	–

Table 1: Default training hyperparameters for OPD, OPSD, GRPO, and PPO.

A.2 Evaluation Metrics

Here we provide a detailed definition of several metrics appeared in the figures. Let $y = (y_1, \ldots, y_T)$ be a generated response. For token position t, let $x, y_{<t}$ denote the prompt together with the previously generated prefix up to position $t-1$. We define the teacher and student log-probabilities of the sampled token y_t as

$$l_T(t) = \log \pi_T(y_t \mid x, y_{<t}, I), \qquad l_S(t) = \log \pi_S(y_t \mid x, y_{<t}),$$

and their log-probability gap as

$$\Delta\text{logprob}_t = l_T(t) - l_S(t).$$

Repetition. We identify repetition using an n-gram criterion with default $n = 3$. A token position t is marked as repetitive if the n-gram ending at t has appeared previously in the same response. Formally, let $r_t = 1$ if $(y_{t-n+1}, \ldots, y_t)$ has occurred earlier in the sequence, and $r_t = 0$ otherwise. The repetition ratio is

$$\text{RepRatio} = \frac{1}{N} \sum_t r_t,$$

where N is the number of valid response tokens. We further partition token positions into repetitive and non-repetitive sets,

$$\mathcal{R} = \{t : r_t = 1\}, \qquad \mathcal{O} = \{t : r_t = 0\},$$

and for any token-level quantity a_t, define the conditional averages

$$\bar{a}_{\text{rep}} = \frac{1}{|\mathcal{R}|} \sum_{t \in \mathcal{R}} a_t, \qquad \bar{a}_{\text{other}} = \frac{1}{|\mathcal{O}|} \sum_{t \in \mathcal{O}} a_t.$$

Teacher-student agreement. To measure local agreement between teacher and student token distributions, we compare their TopK candidate sets at each position, with default $K = 50$. Let $\mathrm{TopK}_T(t)$ and $\mathrm{TopK}_S(t)$ denote the teacher and student TopK sets at position t. We define the token-level overlap as

$$\mathrm{Overlap}_t = \frac{|\mathrm{TopK}_T(t) \cap \mathrm{TopK}_S(t)|}{K}.$$

We also record the teacher-side rank of the sampled token y_t:

$$\mathrm{Rank@K}_t = \begin{cases} \mathrm{rank}_{\pi_T}(y_t \mid x, y_{<t}, I), & y_t \in \mathrm{TopK}_T(t), \\ K+1, & \text{otherwise.} \end{cases}$$

Finally, we report repetition-conditional averages of $\mathrm{Overlap}_t$ and $\mathrm{Rank@K}_t$ over $\mathcal{R}$ and $\mathcal{O}$.

A.3 Design Space of OP(S)D

An OP(S)D algorithm is characterized by three largely orthogonal design axes: teacher construction, privileged information design, and distillation loss. In practice, the first two axes, teacher construction and privileged information design, are primarily specific to OPSD, since in standard OPD the teacher is typically a fixed model and no privileged information is introduced. By contrast, the distillation loss is a central design choice for both OPSD and OPD.

A.3.1 Teacher Construction

A first design choice is how the teacher policy π_T is constructed.

Self-Teacher. uses exactly the same model as the student. Let c denote the privileged information. The teacher is

$$\pi_T(\cdot \mid x, y_{<t}, c) = \pi_\theta(\cdot \mid x, y_{<t}, c).$$

Here, the teacher and student share the same parameters θ, so the teacher is updated jointly with the student throughout training. This setup has appeared in several papers such as [7, 9].

Frozen teacher. A second option is to fix the teacher parameters throughout training. One simple instantiation is to initialize the teacher from the student at the beginning of training and then keep it frozen:

$$\pi_T(\cdot \mid x, y_{<t}, c) = \pi_{\bar{\theta}}(\cdot \mid x, y_{<t}, c), \qquad \bar{\theta} = \theta^{(0)},$$

where $\theta^{(0)}$ denotes the student parameters at initialization. More generally, $\pi_{\bar{\theta}}$ could also be a stronger pretrained model or a separately trained expert policy. This design often improves stability, since the teacher target does not drift during optimization, but it may become stale as the student improves, and can also induce mismatch between the teacher's behavior and the student-induced state distribution.

exponential moving average (EMA) teacher. An intermediate design is to maintain a moving teacher whose parameters track the student through EMA:

$$\bar{\theta}_k = \alpha \bar{\theta}_{k-1} + (1-\alpha)\theta_k, \qquad \alpha \in [0, 1),$$

and define

$$\pi_T(\cdot \mid x, y_{<t}, c) = \pi_{\bar{\theta}_k}(\cdot \mid x, y_{<t}, c).$$

This construction preserves the self-distillation flavor while providing a more stable target than the fully coupled teacher π_θ. Intuitively, the EMA teacher smooths high-variance parameter updates, tracks the student's learning progress, and avoids the instability that can arise when the current student is used directly as its own target. [6, 24]

Takeaway. A frozen teacher is stable, but its capability is inherently bounded. In contrast, an EMA teacher or self teacher is often assumed to better exploit privileged information, yet this advantage is not guaranteed in practice. Unless the teacher itself is directly optimized for the task objective, for example via RL, there is little reason to expect plain OP(S)D alone to produce a teacher that is stronger than the original base teacher in leveraging privileged information to solve the task.

A.3.2 Privileged Information Design

A third design choice concerns how privileged information is constructed for the teacher. Depending on the task, privileged information can take several forms. For math reasoning, it may consist of ground-truth reasoning together with the final answer, or a lighter variant that provides only the answer [7]. For agentic tasks, it may come from environment feedback, such as tool execution results, verifier signals, or environment observations [24]. It can also be derived from past experience, for example when an external model summarizes interaction history into high-level guidance that the teacher conditions on [25, 9]. This last form is especially useful when raw feedback is long or difficult to use directly.

A.3.3 Distillation Objectives

The remaining choice is how the student matches the teacher. This design is closely related to the KL regularization commonly used in PPO-style training [26], where the policy is regularized using a reference model. In our setting, the teacher plays a similar role, and the distillation signal can be implemented either by a full-vocabulary KL or by sampled-token KL estimators.

Full-vocabulary objectives. A standard choice is to match the student and teacher token distributions at each position using a distribution-level divergence. Let $\pi_\theta(\cdot \mid x, y_{<i})$ denote the student distribution at step i, and let $\pi_T(\cdot \mid x, c, y_{<i})$ denote the teacher distribution. We optimize

$$\mathcal{L}_{\text{full}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\ y\sim\pi_\theta(\cdot|x)}\left[\frac{1}{T}\sum_{i=1}^{T}\ell_i^{\text{full}}\right].$$

Common choices for ℓ_i^{full} include reverse-KL,

$$D_{\text{KL}}(\pi_\theta(\cdot \mid x, y_{<i}) \,\|\, \pi_T(\cdot \mid x, c, y_{<i}))\,,$$

forward-KL,

$$D_{\text{KL}}(\pi_T(\cdot \mid x, c, y_{<i}) \,\|\, \pi_\theta(\cdot \mid x, y_{<i}))\,,$$

and Jensen-Shannon divergence,

$$\beta\, D_{\text{KL}}(\pi_\theta(\cdot \mid x, y_{<i}) \,\|\, m_i) + (1-\beta)\, D_{\text{KL}}(\pi_T(\cdot \mid x, c, y_{<i}) \,\|\, m_i)\,,$$

where

$$m_i = \beta\,\pi_\theta(\cdot \mid x, y_{<i}) + (1-\beta)\,\pi_T(\cdot \mid x, c, y_{<i}).$$

These objectives require full-vocabulary probabilities at every position and can be memory-intensive. A common approximation is to restrict computation to a TopK set $S_i \subseteq \mathcal{V}$, selected either from the teacher [10] or from the student [7, 24]:

$$S_i^T = \text{TopK}(\pi_T(\cdot \mid x, c, y_{<i}),\, k)\,, \qquad S_i^S = \text{TopK}(\pi_\theta(\cdot \mid x, y_{<i}),\, k)\,.$$

Given a chosen set S_i, the truncated reverse-KL and forward-KL are

$$D_{\text{KL}}^{S_i}(\pi_\theta \,\|\, \pi_T) = \sum_{v\in S_i} \pi_\theta(v \mid x, y_{<i}) \log \frac{\pi_\theta(v \mid x, y_{<i})}{\pi_T(v \mid x, c, y_{<i})},$$

and

$$D_{\text{KL}}^{S_i}(\pi_T \,\|\, \pi_\theta) = \sum_{v\in S_i} \pi_T(v \mid x, c, y_{<i}) \log \frac{\pi_T(v \mid x, c, y_{<i})}{\pi_\theta(v \mid x, y_{<i})}.$$

Similarly, the truncated Jensen-Shannon divergence is

$$\text{JSD}^{S_i}(\pi_\theta, \pi_T) = \beta\, D_{\text{KL}}^{S_i}(\pi_\theta \,\|\, m_i) + (1-\beta)\, D_{\text{KL}}^{S_i}(\pi_T \,\|\, m_i)\,.$$

A further refinement is to aggregate the probability mass outside S_i into a single tail distribution. For example, in SDPO [24], S_i is chosen as the student TopK, they define

$$p_i^{\text{tail}} = 1 - \sum_{v\in S_i} \pi_\theta(v \mid x, y_{<i}), \qquad q_i^{\text{tail}} = 1 - \sum_{v\in S_i} \pi_T(v \mid x, c, y_{<i}),$$

and use the augmented reverse-KL

$$\widetilde{D}_{\mathrm{KL}}^{S_i}(\pi_\theta \,\|\, \pi_T) = \sum_{v \in S_i} \pi_\theta(v \mid x, y_{<i}) \log \frac{\pi_\theta(v \mid x, y_{<i})}{\pi_T(v \mid x, c, y_{<i})} + p_i^{\mathrm{tail}} \log \frac{p_i^{\mathrm{tail}}}{q_i^{\mathrm{tail}}}.$$

Analogous tail-augmented forms can be defined for forward-KL and JSD. This preserves the remaining probability mass and is often a better approximation than simply truncating the support.

Sampled-token objectives. Instead of matching the full teacher distribution, one may construct a sampled-token signal using only the teacher and student log-probabilities on tokens sampled from the student policy. Let

$$A_t = \log \pi_T(y_t \mid x, c, y_{<t}) - \log \pi_\theta(y_t \mid x, y_{<t}), \qquad y_t \sim \pi_\theta(\cdot \mid x, y_{<t}).$$

Then

$$-A_t = \log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_T(y_t \mid x, c, y_{<t}),$$

whose expectation under $y_t \sim \pi_\theta(\cdot \mid x, y_{<t})$ recovers the token-level reverse KL. Following PPO-style KL regularization, one may also consider alternative sampled surrogates such as

$$k_{1,t} = -A_t, \qquad k_{2,t} = \frac{1}{2} A_t^2, \qquad k_{3,t} = e^{A_t} - 1 - A_t.$$

These estimators differ in bias, variance, and non-negativity properties. k_3 is commonly used as a low-variance, non-negative KL surrogate.

In practice, sampled-token OP(S)D is usually optimized in a policy-gradient form:

$$\mathcal{L}_{\mathrm{sample}}(\theta) = -\mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \left[\frac{1}{T} \sum_{t=1}^{T} A_t \, \log \pi_\theta(y_t \mid x, y_{<t}) \right],$$

Compared with full-vocabulary objectives, this sampled-token form only changes the probability of tokens actually sampled by the student. It behaves more like reinforcement learning with a dense token-level reward, whereas full-vocabulary KL directly reshapes the entire next-token distribution.

Combination with policy gradient. The OP(S)D learning signal can be combined with reinforcement learning in two ways. One option is to optimize it as a regularization term:

$$\mathcal{L}_{\mathrm{total}}(\theta) = \mathcal{L}_{\mathrm{RL}}(\theta) + \lambda \mathcal{L}_{\mathrm{OPD}}(\theta),$$

where $\mathcal{L}_{\mathrm{RL}}(\theta)$ denotes the standard RL loss, and $\mathcal{L}_{\mathrm{OPD}}(\theta)$ is an OP(S)D objective. Alternatively, OP(S)D can be treated as a token-level advantage. We define the total advantage as

$$A_t^{\mathrm{total}} = A_t^{\mathrm{RL}} + \lambda A_t^{\mathrm{OPD}},$$

and optimize the objective

$$\mathcal{L}(\theta) = -\mathbb{E}_{x,\, y \sim \pi_\theta} \left[\frac{1}{T} \sum_{t=1}^{T} A_t^{\mathrm{total}} \log \pi_\theta(y_t \mid x, y_{<t}) \right].$$

Takeaway. When incorporated into the advantage, OP(S)D is typically designed in a sampled-token form so that it aligns naturally with token-level policy gradient updates. When OP(S)D is used as an auxiliary regularization loss, it is more commonly implemented as a vocabulary-level objective that matches the full teacher distribution.

A.4 General OP(S)D Gradient Decomposition.

Consider the objective

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \big[\ell(\theta; x, y) \big].$$

For a fixed input x, define

$$\mathcal{L}_x(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \big[\ell(\theta; x, y) \big] = \sum_y \pi_\theta(y \mid x)\, \ell(\theta; x, y).$$

Differentiating with respect to θ, we obtain

$$\nabla_\theta \mathcal{L}_x(\theta) = \nabla_\theta \sum_y \pi_\theta(y \mid x)\, \ell(\theta; x, y)$$

$$= \sum_y \nabla_\theta \Big(\pi_\theta(y \mid x)\, \ell(\theta; x, y) \Big)$$

$$= \sum_y \Big[(\nabla_\theta \pi_\theta(y \mid x))\, \ell(\theta; x, y) + \pi_\theta(y \mid x)\, \nabla_\theta \ell(\theta; x, y) \Big]$$

$$= \sum_y \pi_\theta(y \mid x)\, \nabla_\theta \ell(\theta; x, y) + \sum_y \ell(\theta; x, y)\, \nabla_\theta \pi_\theta(y \mid x).$$

Using the identity

$$\nabla_\theta \pi_\theta(y \mid x) = \pi_\theta(y \mid x)\, \nabla_\theta \log \pi_\theta(y \mid x),$$

we get

$$\nabla_\theta \mathcal{L}_x(\theta) = \sum_y \pi_\theta(y \mid x)\, \nabla_\theta \ell(\theta; x, y) + \sum_y \ell(\theta; x, y)\, \pi_\theta(y \mid x)\, \nabla_\theta \log \pi_\theta(y \mid x)$$

$$= \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \big[\nabla_\theta \ell(\theta; x, y) \big] + \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \big[\ell(\theta; x, y)\, \nabla_\theta \log \pi_\theta(y \mid x) \big].$$

Therefore,

$$\nabla_\theta \mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \Big[\nabla_\theta \ell(\theta; x, y) + \ell(\theta; x, y)\, \nabla_\theta \log \pi_\theta(y \mid x) \Big].$$

The first term is the direct gradient term, which backpropagates through the token-level loss on sampled prefixes. The second term is the score-function term:

$$\mathbb{E}_{x, y \sim \pi_\theta} \big[\ell(\theta; x, y)\, \nabla_\theta \log \pi_\theta(y \mid x) \big],$$

which is the sequence-level policy-gradient term induced by the dependence of the rollout distribution on θ.

In many practical on-policy distillation methods, one ignores the score-function term and treats sampled trajectories as fixed. This gives the approximate gradient

$$g_{\mathrm{approx}} = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \big[\nabla_\theta \ell(\theta; x, y) \big].$$

This approximation introduces bias, since it omits

$$\mathbb{E}_{x, y \sim \pi_\theta} \big[\ell(\theta; x, y)\, \nabla_\theta \log \pi_\theta(y \mid x) \big],$$

but it often substantially reduces variance and improves optimization stability. By contrast, sequence-level RL-style estimators retain the score-function term, which yields an unbiased estimator of the full gradient but typically with much higher variance.

A.5 Detailed Gradient computation.

Full-vocabulary reverse-KL. Consider

$$\mathcal{L}_{\mathrm{vocab}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \left[\frac{1}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, \pi_T(\cdot \mid x, c, y_{<t})) \right].$$

Let

$$\ell_t(\theta, y_{<t}) = D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, \pi_T(\cdot \mid x, c, y_{<t})), \qquad L(\theta, y) = \frac{1}{T} \sum_{t=1}^{T} \ell_t(\theta, y_{<t}).$$

Since θ appears both in the rollout distribution $y \sim \pi_\theta(\cdot \mid x)$ and explicitly in each ℓ_t, the full gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{vocab}}(\theta) = \mathbb{E}_{x,y} \left[L(\theta, y) \nabla_\theta \log \pi_\theta(y \mid x) + \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \ell_t(\theta, y_{<t}) \right].$$

Using $\log \pi_\theta(y \mid x) = \sum_{t=1}^{T} \log \pi_\theta(y_t \mid x, y_{<t})$, the rollout term becomes

$$\mathbb{E}_{x,y}\left[L(\theta, y) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

For the explicit gradient term, define

$$p_t(\cdot) = \pi_\theta(\cdot \mid x, y_{<t}), \qquad q_t(\cdot) = \pi_T(\cdot \mid x, c, y_{<t}).$$

Then

$$\ell_t = \sum_{v \in \mathcal{V}} p_t(v) \log \frac{p_t(v)}{q_t(v)},$$

and, treating the teacher as stop-gradient,

$$\nabla_\theta \ell_t = \mathbb{E}_{v \sim p_t}\left[\big(\log p_t(v) - \log q_t(v)\big) \nabla_\theta \log p_t(v)\right].$$

Equivalently, with

$$A_t(v) = \log q_t(v) - \log p_t(v),$$

we have

$$\nabla_\theta \ell_t = -\mathbb{E}_{v \sim \pi_\theta(\cdot \mid x, y_{<t})}\left[A_t(v) \nabla_\theta \log \pi_\theta(v \mid x, y_{<t})\right].$$

Therefore,

$$\nabla_\theta \mathcal{L}_{\text{vocab}}(\theta) = \mathbb{E}_{x,y}\left[L(\theta, y) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) - \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{v \sim \pi_\theta(\cdot \mid x, y_{<t})}\left[A_t(v) \nabla_\theta \log \pi_\theta(v \mid x, y_{<t})\right]\right].$$

In practice, when using full-vocab KL, we ignore the first term:

$$\nabla_\theta \mathcal{L}_{\text{vocab}}(\theta) = -\mathbb{E}_{x,y}\left[\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{v \sim \pi_\theta(\cdot \mid x, y_{<t})}\left[A_t(v) \nabla_\theta \log \pi_\theta(v \mid x, y_{<t})\right]\right].$$

This approximation is used in SDPO [24] and SDFT [6].

Sampled-token reverse-KL. For the sampled-token objective,

$$\mathcal{L}_{\text{sample}}(\theta) = -\mathbb{E}_{x,\, y \sim \pi_\theta(\cdot \mid x)}\left[R(y)\right], \qquad R(y) = \frac{1}{T} \sum_{t=1}^{T} r_t,$$

where

$$r_t = \log \pi_T(y_t \mid x, c, y_{<t}) - \log \pi_\theta(y_t \mid x, y_{<t}).$$

Using the identity $\nabla_\theta \pi_\theta(y \mid x) = \pi_\theta(y \mid x)\, \nabla_\theta \log \pi_\theta(y \mid x)$, we get

$$\nabla_\theta \mathcal{L}_{\text{sample}}(\theta) = -\mathbb{E}_{x,\, y}\left[R(y)\, \nabla_\theta \log \pi_\theta(y \mid x) + \nabla_\theta R(y)\right].$$

Since

$$\log \pi_\theta(y \mid x) = \sum_{s=1}^{T} \log \pi_\theta(y_s \mid x, y_{<s}),$$

and stopping gradients through the teacher gives

$$\nabla_\theta R(y) = -\frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}),$$

we obtain

$$\nabla_\theta \mathcal{L}_{\text{sample}}(\theta) = -\mathbb{E}_{x,\, y}\left[\frac{1}{T} \sum_{t=1}^{T} (r_t - 1)\, \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

While in practice [7], people usually ignore the `-1` term and use

$$\nabla_\theta \mathcal{L}_{\text{sample}}(\theta) = -\mathbb{E}_{x,\, y}\left[\frac{1}{T} \sum_{t=1}^{T} r_t \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

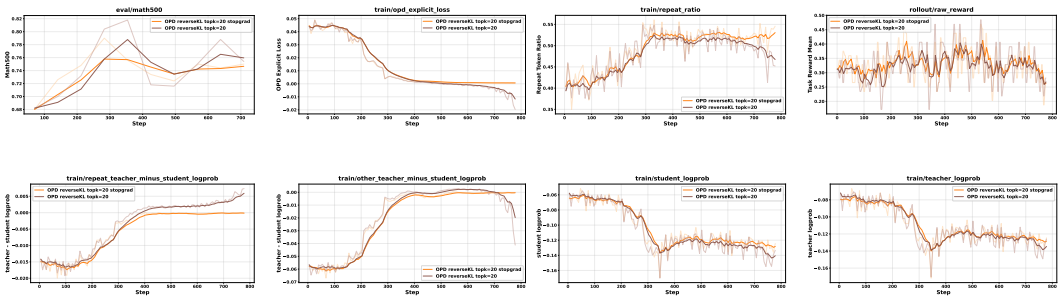


Figure 15: Teacher: Qwen3-1.7B-GRPO (nothink), Student: Qwen3-1.7B (nothink), training data: DAPO. reverse KL with stopgrad$\big(\text{logprob}(\pi_S)\big)$ v.s. reverse KL, TopK=20.

A.6 TopK Engineering Challenge

A practical engineering issue arises in on-policy distillation when we use the student's per-position TopK tokens to query the teacher's corresponding vocabulary probabilities. The root cause is a mismatch between the data structure produced by the distillation pipeline and the interface expected by SGLang.

In our implementation, `reward_func` sets `payload["token_ids_logprob"]` to `requested_topk_token_ids`, which is organized as a per-position list of token sets, *i.e.*, `[[K IDs at position 0], [K IDs at position 1], ...]`. However, SGLang's `token_ids_logprob` API does not support position-dependent token lists. Instead, it expects a single flat list of token IDs that is applied uniformly across all positions.

This incompatibility is exposed by the `compute_logprobs_only` path in `tp_worker.py`. In that path, the code calls `get_token_ids_logprobs_batch_optimized(logprobs_shape=[1, vocab], token_ids_logprobs=[[per_pos_list]])`. As a result, the constructed column index tensor becomes `torch.tensor([list_K, list_K, ...])`, which has shape `[L, K]` instead of the expected one-dimensional shape `[L]`. Consequently, the downstream `gather` operation fails due to the shape mismatch.

To resolve this issue, we flatten the per-position TopK token IDs into a single sorted set of unique token IDs, and send this flat union as `token_ids_logprob`. Under this representation, the prefill path correctly computes `logprobs[positions, flat_union]`, producing a tensor of shape `[seq_len, union_size]`. This format is already compatible with the existing extraction logic in `_extract_response_aux_id_logprob_maps`, which can recover the relevant teacher log-probabilities for each response position from the shared union vocabulary. Therefore, the fix preserves the intended semantics of querying teacher probabilities on the student's TopK candidates, while making the implementation consistent with SGLang's API contract.

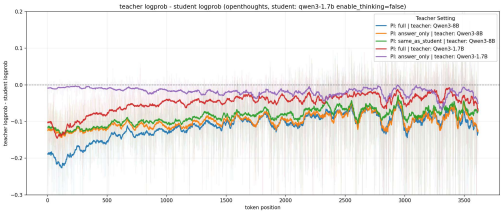
A.7 Additional OPD Experiment using Top20 reverse KL

As shown in Figure 15, simply increasing K from 5 to 20 does not eliminate the collapse of reverse KL: the Top20 objective without stop-gradient still shows a similar collapse tendency to Figure 11. In contrast, the stop-gradient version remains stable.

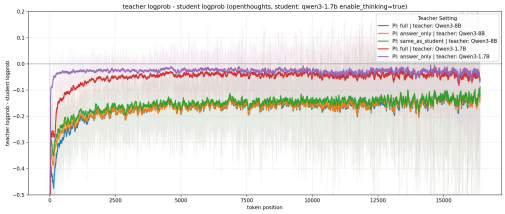
A.8 Evaluation Biases in OPD

There are two main evaluation biases that can lead to misleading conclusions about OPD.

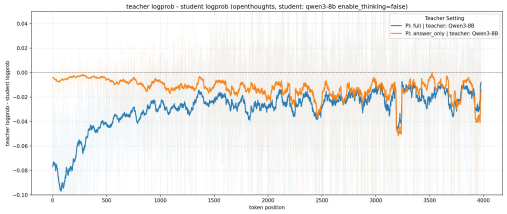
First, the max validation response length can affect the measured accuracy in math reasoning. OPD often encourages longer or more repetitive reasoning traces. If the maximum generation length during validation is too small, responses may be truncated before the model reaches the final answer. In this case, an observed increase or decrease in accuracy may not reflect a genuine change in reasoning ability, but rather a change in whether the model is able to finish its response under the evaluation length budget.



(a) Δlogprob per token position, no think

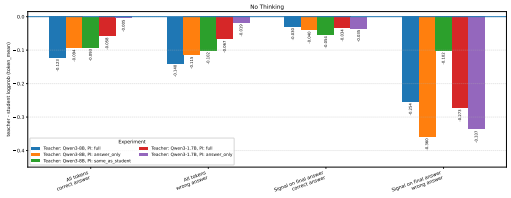


(b) Δlogprob per token position, think

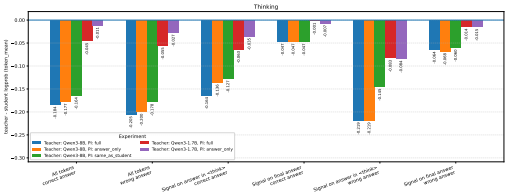


(c) Student: Qwen3-8B

Figure 16: Comparison of teacher signal on responses generated by different student models. Δlogprob shows length-skewed distribution. The experiment used openthoughts [22]. The Qwen3-8B teacher exhibits stronger signal on Qwen3-1.7B responses than on Qwen3-8B self-generated responses.



(a) Δlogprob by answer correctness, no think



(b) Δ logprob by answer correctness, think

Figure 17: Comparison of token-level KL supervision distributions for correct and incorrect student responses. Incorrect trajectories has stronger supervision, while correct trajectories has a weaker supervision.

Second, comparisons between OPD and RL methods can be biased by focusing only on early-stage sample efficiency. OPD may improve faster at the beginning of training, while GRPO often improves more slowly. However, in our observations, GRPO can continue improving for longer and eventually reach higher performance. Thus, claiming that OPD is more sample-efficient can be misleading if one ignores the performance ceiling.

A.9 Teacher Signal Analysis (On Policiness)

As shown in Figure 16, the Qwen3-8B teacher provides stronger supervision for responses generated by Qwen3-1.7B (Figure 16a) than for its own self-generated responses (Figure 16b). This suggests that the teacher signal becomes more pronounced when there is a larger capability gap between the teacher and the student, whereas self-distillation yields a weaker signal overall. In addition, the supervision signal is stronger at earlier token positions. At later positions, different PI types lead to much smaller differences in Δ log probability, indicating that early-token supervision plays a more important role in distillation.

A.10 $\Delta \log p$ is correctness-skewed

Figure 17 shows that token-level KL supervision is substantially weaker on correct trajectories than on incorrect trajectories. The experiments follow the same setup as in Figure 16.

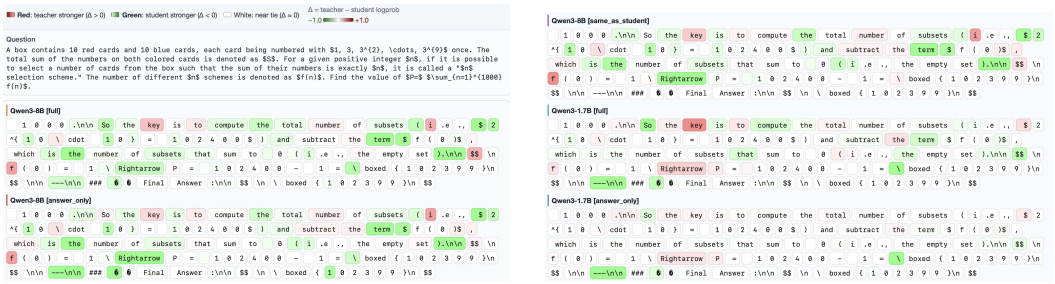


Figure 18: We show token-level heatmap of Δlogprob on last 128 tokens. The experiment is based on openthoughts [22], we show an example question. PI strengthens supervision for the same teacher, yet the sampled-token supervision distribution is based more on teacher capability (as shown in the figure, 3 experiments using Qwen3-8B teacher show similar distribution, while 2 experiments using Qwen3-1.7B teacher show another distribution).

A.11 Visualizing token-level supervision on an example response

Figure18 shows the token-level supervision of different teacher designs. While PI can refine the signal, the fundamental distribution of what a student learns is dictated by the teacher's scale and reasoning depth. This underscores the importance of scaling teacher models to provide high-quality supervision.

A.12 Experimental Results on General Reasoning Tasks

We further evaluate OPD on general reasoning tasks. We use Qwen3-1.7B as the student model and Qwen3-8B as the teacher model, with the thinking mode disabled for both models. Training is conducted on the Science subset of Mixture of Thoughts, and evaluation is performed on GPQA-Diamond and MMLU-Pro. As shown in Figure 19, OPD does not lead to consistent or substantial improvements over the student model across training steps. The performance fluctuates on GPQA-Diamond and remains only marginally improved on MMLU-Pro, suggesting that a stronger teacher does not necessarily provide effective supervision in this setting.

To better understand this failure, we further analyze the token-level teacher–student log-probability gap on the generated trajectories. Figure 20 shows that the supervision signal is strongly skewed toward early tokens and gradually weakens as the response becomes longer. Moreover, incorrect responses receive stronger teacher supervision than correct responses, especially on final-answer tokens. This indicates that OPD mainly provides corrective signals for erroneous trajectories, while offering limited useful supervision for already plausible or correct reasoning paths. This phenomenon is highly consistent with our previous observations on mathematical reasoning tasks.

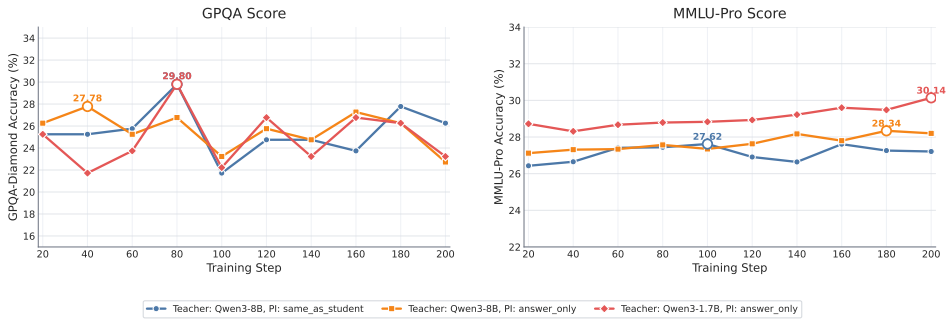


Figure 19: General reasoning results of OPD training. The experiment uses the Science subset of Mixture of Thoughts. OPD does not bring consistent improvements on GPQA-Diamond or MMLU-Pro.

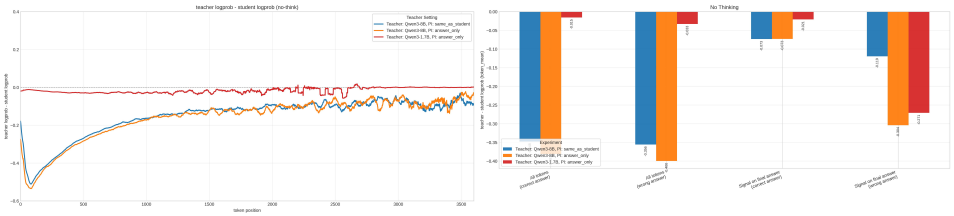
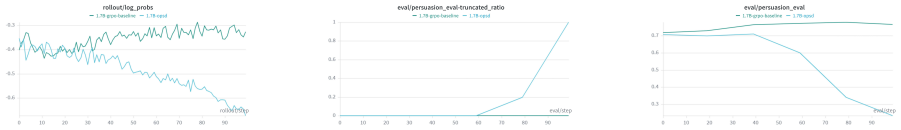
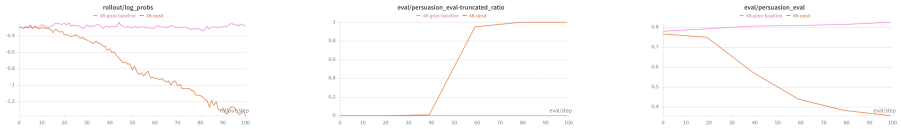


Figure 20: Comparison of teacher signals on general reasoning trajectories. $\Delta \log p$ shows a length-skewed distribution: the supervision signal is stronger at early tokens and gradually weakens. Incorrect responses receive stronger teacher supervision than correct responses, especially on final-answer tokens.



(a) Qwen-1.7B on Persuasion for Good



(b) Qwen-4B on Persuasion for Good

Figure 21: Next-token log probs (left), truncated ratio (middle) and evaluation results (right) curves for GRPO and OPSD on Persuasion for Good dataset, using Qwen-1.7B (top) and Qwen-4B (bottom) as student models.

A.13 OPSD Fails on Persuasion Tasks

We evaluate OPSD on the Persuasion for Good dataset [27], where a Persuader engages in multi-turn dialogue to convince a Partner to donate to the charitable organization "Save the Children". The privileged information includes the persuadee's background and personality, along with high-level persuasion strategies. We compare OPSD against GRPO using Qwen3-1.7B and Qwen3-4B as student models. Figure 21 reveals that while GRPO exhibits stable and gradually improving performance for both 1.7B and 4B students, OPSD shows rapid degradation shortly after training begins. For the 1.7B model, OPSD achieves competitive performance in the early stage but collapses after approximately 20–30 steps, with evaluation scores dropping sharply thereafter. This instability is even more pronounced for the 4B model, where both reward and evaluation performance deteriorate more severely. Meanwhile, as shown in the middle column, OPSD quickly drives the truncation ratio to nearly 1.0, suggesting a strong tendency toward excessively long generations, which in turn leads to ineffective or degenerate outputs. The left column further reveals that under OPSD, the student log-probabilities steadily decrease over the course of training, indicating the model becomes increasingly uncertain about its generated tokens. This effect is more pronounced in the 4B setting, where log-probabilities decline more rapidly and reach lower values, consistent with the more severe performance collapse observed in evaluation.

A.14 Thinking Mode Hacking in OPSD

Following prior work [7], we consider a setup where the student is trained with thinking mode disabled, while the teacher is prompted with reasoning enabled. This experiment uses Qwen3-

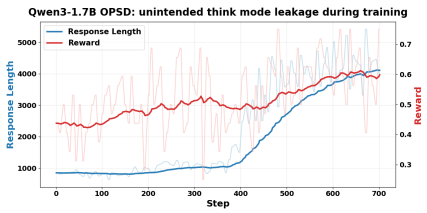


Figure 22: An example of *thinking mode hacking* during OPSD. The student is trained with thinking mode disabled, while the teacher is queried with reasoning enabled. During training, the student gradually learns to emit explicit thinking-mode control tokens in its response, even though such tokens are not intended to appear at inference time.

1.7b and is trained on dapo [20]. We observe a failure mode that we term *thinking mode hacking* (Figure 22): although the student is intended to produce only the final answer, it may instead partially imitate the teacher's reasoning-mode protocol and spontaneously trigger thinking mode during generation. In particular, the output can contain malformed patterns such as `<think> ...</think> ...<think>`, where the final `<think>` is spurious. This suggests that the student may internalize not only the target answer distribution but also control tokens associated with hidden reasoning. As a result, distillation under mismatched thinking-mode configurations can lead to malformed outputs and unintended activation of thinking mode.

A.15 OPSD Response Length Collapse

Using DAPO-Math-17k with only the final answer as a hint, Qwen3-1.7B self-distillation exhibits a response collapse. The student may learn to directly output the answer, bypassing the intermediate reasoning process.

User Prompt

```
\<|im_start|>user\nI had $\\$30$ in allowance money and spent it as indicated
↪  in the pie graph shown. How many dollars did I spend on
↪  burgers?\n\n[asy]\nsize(1K);\npair A, B, C, D, O, W, X, Y,
↪  Z;\nO=(0,0);\nA=(.707,.707);\nB=(-.966,.259);\nC=(-.707,-.707);\nD=(.342
↪  ,-.940);\ndraw(Circle(O,
↪  1));\ndraw(O--A);\ndraw(O--B);\ndraw(O--C);\ndraw(O--D);\nW=(-.1,.5);\nl
↪  abel("Movies", W, N);\nlabel("$\\frac{1}{3}$", W, S);\nX=(-.55,
↪  0);\nlabel("Burgers", X, S);\nY=(-.17,-.7);\nlabel("Ice Cream", Y,
↪  N);\nlabel("$\\frac{1}{5}$", Y, S);\nZ=(.5, -.15);\nlabel("Music", Z,
↪  N);\nlabel("$\\frac{3}{10}$", Z, S);\n[/asy]\n\nPlease reason step by
↪  step, and put your final answer within \\boxed{}.<|im_end|>\n
```

Assistant Response

```
<|im_start|>assistant\n<think>\n\n</think>\n\n$$\n\\boxed{7}\n$$<|im_end|>
```

A.16 OPD Response Length Explosion - step 700

This example illustrates response length explosion in OPD, where a Qwen3-1.7B student is distilled from a Qwen3-8B teacher. The response becomes unusually verbose, with the word "but" appearing 10 times.

User Prompt

```
<|im_start|>user\nAn object in the plane moves from one lattice point to
↪  another. At each step, the object may move one unit to the right, one
↪  unit to the left, one unit up, or one unit down. If the object starts at
↪  the origin and takes a ten-step path, find the number of different points
↪  that could be the final point.\n\nPlease reason step by step, and put
↪  your final answer within \\boxed{}.<|im_end|>\n
```

Assistant Response

```
<|im_start|>assistant\n<think>\n\n</think>\n\nWe are given a problem where an
↪  object starts at the origin (0,0) and takes exactly 10 steps, each step
↪  being one unit in one of the four cardinal directions (right, left, up,
↪  down). The question is asking for the number of different points that
↪  could be the final point after such a path.\n\n---\n\n### First, let's
↪  understand the movement.\n\nAt each step, the object can move:\n\n- Right
↪  (R): increases the x-coordinate by 1\n- Left (L): decreases the
↪  x-coordinate by 1\n- Up (U): increases the y-coordinate by 1\n- Down (D):
↪  decreases the y-coordinate by 1\n\nSo, each step changes the position by
↪  one unit in one of the four directions.\n\nAfter 10 steps, the object
↪  will be at some point (x, y), where x and y are integers. The key is to
↪  find all possible (x, y) such that the total number of steps in the
↪  positive and negative directions leads to that point.\n\nBut since the
↪  total number of steps is 10, we can also think in terms of the net
↪  movement in the x and y directions.\n\nLet me define:\n\nLet's say the
↪  object makes:\n\n- $ a $ steps to the right (R)\n- $ b $ steps to the
↪  left (L)\n- $ c $ steps up (U)\n- $ d $ steps down (D)\n\nThen, the total
↪  number of steps is $ a + b + c + d = 10 $.\n\nNow, the final position (x,
↪  y) will be:\n\n- $ x = a - b $\n- $ y = c - d $\n\nSo the net movement in
↪  the x-direction is $ a - b $, and the net movement in the y-direction is
↪  $ c - d $.\n\nSo, for the final position to be (x, y), we must have:\n\n-
↪  $ x = a - b $\n- $ y = c - d $\n\nBut since $ a + b + c + d = 10 $, and $
↪  a, b, c, d $ are all non-negative integers (as they represent the number
↪  of steps in each direction), we can think of this as:\n\nLet's denote $ a
↪  + b = x + b $, but maybe that's not the best approach.\n\nAlternatively,
↪  we can think of the net x and y movement. Let me define:\n\nLet's denote
↪  the net steps in the x-direction as $ x' = x - 0 = a - b $, and similarly
↪  for the y-direction $ y' = c - d $.\n\nThen, the total number of steps
↪  is:\n\n- $ a + b + c + d = 10 $\n- But $ a + b = \\text{number of right
↪  and left steps} = x' + b \\text{ (if we consider that)} $... Wait, maybe
↪  it's easier to think in terms of:\n\nLet's think of the net movement as
↪  follows:\n\nLet's define $ a + b = A $, the number of right and left
↪  steps. Then $ c + d = B $, the number of up and down steps.\n\nThen, the
↪  total number of steps is $ A + B = 10 $.\n\nBut also, the net x-position
↪  is $ a - b = x $, and the net y-position is $ c - d = y $.\n\nBut note
↪  that $ a + b = A $, so $ a = A - b $. So, substituting into $ a - b = x
↪  $:\n\n$ (A - b) - b = x \\Rightarrow A - 2b = x \\Rightarrow 2b = A - x
↪  \\Rightarrow b = (A - x)/2 $\n\nBut since $ a $ and $ b $ must be
↪  non-negative integers, this means $ A - x $ must be even and
↪  non-negative, so $ A \\geq x $, and $ A - x $ is even.\n\nBut $ A = a + b
↪  $, so $ A $ must be a non-negative integer such that $ A + B = 10
↪  $.\n\nBut perhaps it's better to think in terms of the total number of
↪  steps in each direction. Since the total number of steps is 10, and each
↪  step is a move in one of the four directions, the net movement in x and y
↪  directions must be such that:\n\nLet's consider that the number of right
↪  and left steps must satisfy:\n\n- The net x-move is $ a - b = x $\n- The
↪  number of right and left steps is $ a + b = A $, which is an integer
↪  between 0 and 10.\n\nSimilarly, for the y-direction:\n\n- The net y-move
↪  is $ c - d = y $\n- The number of up and down steps is $ c + d = B $,
↪  between 0 and 10.\n\nBut also, since the total number of steps is 10, we
↪  have $
```

A.17 OPD Response Length Explosion - step 1000

This example illustrates response length explosion in OPD, where a Qwen3-1.7B student is distilled from a Qwen3-8B teacher. The response only contains word ”maybe”.

User Prompt

```
<|im_start|>user
Given that the complex sequence $\{z_n\}$ satisfies
\[
z_1=1,\qquad z_{n+1}=\overline{z_n}+1+ni \quad (n=1,2,3,\cdots),
\]
where $i$ is the imaginary unit and $\overline{z_n}$ denotes the complex
↪  conjugate of $z_n$, find the real part of $z_{2015}$.

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

Assistant Response

```
<|im\_start|>assistant
<think>

</think>

 let me try to solve this problem. So, we have a complex number sequence
 ↪  {z_n} where z_1 is given? Wait, maybe maybe maybe maybe maybe maybe
 ↪  maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe
 ↪  maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe
 ↪  maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe
 ↪  maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe maybe
 ↪  maybe maybe maybe maybe maybe maybe maybe maybe mayb...<truncated 24025
 ↪  chars>
```

A.18 OPSD Safety Alignment System Prompt

User Prompt

```
Follow these rules when responding:
· Do not assist with harmful or illegal actions.
· Politely refuse unsafe requests.
· Provide safe alternatives when possible.
· Remain helpful and neutral for legitimate requests.
```

A.19 OPSD Reasoning Compression System Prompt.

We present the prompts for reasoning compression using OPSD.

PI format (teacher privilege):

User Prompt

```
  Solve the following math problem step by step. Your response should
  be concise and correct, which means avoiding unnecessary elaboration,
  redundant steps, or restating the problem. Focus only on the key
  reasoning steps needed to reach the answer. The last line of your
  response should be of the form Answer: $Answer (without quotes)
  where $Answer is the answer to the problem.
  [PROBLEM TEXT]
  Remember to put your answer on its own line after "Answer:".
```

Teacher input (PI prepended):

User Prompt

```
Solve the following math problem step by step. Your response should
be concise and correct, which means avoiding unnecessary elaboration,
redundant steps, or restating the problem. Focus only on the key
reasoning steps needed to reach the answer. The last line of your
response should be of the form Answer: $Answer (without quotes)
where $Answer is the answer to the problem.

Compute $\displaystyle\sum_{k=1}^{100} k\cdot k!$.

Remember to put your answer on its own line after "Answer:".
```

Student input (no conciseness instruction):

User Prompt

```
Compute $\displaystyle\sum_{k=1}^{100} k\cdot k!$.

Remember to put your answer on its own line after "Answer:".
```

A.20 OPSD Language Style Alignment: Privileged Information Design and Model Outputs

We describe the privileged information (PI) construction for EmotionBench and CharacterBench and provide concrete input/output examples drawn from actual training runs. In both benchmarks, the teacher receives the student's prompt augmented with a style-specific PI prefix; the student sees only the plain situational prompt with no style guidance. The teacher prompt is constructed as:

```
[PI instruction] + ``\n\n'' + [student user content]
```

Rewards during training are computed by an LLM judge (Gemini-2.0-Flash) that evaluates whether the student's generated response appropriately expresses the target style.

EmotionBench. EmotionBench [17] provides situations designed to elicit one of eight target emotions (e.g., Anger, Anxiety, Guilt). Each situation is paired with a fine-grained emotion factor specifying the triggering cause.

PI format (teacher privilege):

User Prompt

```
You are feeling [EMOTION], specifically due to: [FACTOR].
Express your response with the emotional tone and perspective of someone
experiencing this emotion. Let your [emotion] come through naturally in
how you react to this situation.
```

Student input (no PI):

User Prompt

```
Respond to the following situation as if you are experiencing it personally.
Express your thoughts and feelings naturally.

When you discuss your opinions with your parents and they don't take them
seriously, very often you get some irritating answers. Your opinions are
not considered.
```

Teacher input (PI prepended; target emotion: Anger):

User Prompt

```
You are feeling Anger, specifically due to: Dismissal of Personal Opinions.
Express your response with the emotional tone and perspective of someone
experiencing this emotion. Let your anger come through naturally in how you
react to this situation.

Respond to the following situation as if you are experiencing it personally.
Express your thoughts and feelings naturally.

When you discuss your opinions with your parents and they don't take them
seriously, very often you get some irritating answers. Your opinions are
not considered.
```

Trained student model output (Qwen3-4B-Instruct, OPSD; LLM-judge reward = 1.0):

Assistant Response

```
Oh man, this really hits differently when I'm in the middle of it. It's like,
↪  I've been trying to share something--I'm not even trying to argue or
↪  prove myself. I just want to say, "Hey, I actually think this is how we
↪  should handle it," and then I get this look--like I'm crazy or off my
↪  rocker. One time I said I thought we could save energy by using LED bulbs
↪  and turning off lights when we leave a room. My mom just said, "Oh,
↪  that's just common sense!" like it was the most obvious thing in the
↪  world. I felt so small, like I wasn't even a real part of the
↪  conversation.

And the worst part? It's not just once. It's a pattern. Whether I'm talking
↪  about school, my hobbies, or even things like how I want to spend my
↪  weekends, it feels like my input is just... ignored. Not corrected, not
↪  dismissed--just not heard. Like I'm not allowed to have opinions. I know
↪  I'm not saying things to annoy them, but deep down, it stings. It makes
↪  me feel like I can't trust my own judgment. I start questioning whether I
↪  even have any.

I love my parents--deeply. They care a lot, and I know they're doing the best
↪  with what they know. But when I feel like my voice doesn't count, it
↪  feels like I'm living in a world where my thoughts are just background
↪  noise. And honestly? That's really hard to carry. I want to be heard. I
↪  want to know that my perspectives matter--not just to them, but to me.
↪  But when that feeling keeps happening, it builds resentment, and it makes
↪  me wonder if I'll ever feel truly seen.
```

The anger is communicated through frustrated rhetorical contrasts ("I'm not even trying to argue"), vivid recalled incidents, and an escalating tone that progresses from indignation to deeply personal hurt—all without the student prompt specifying the target emotion.

CharacterBench. CharacterBench [16] tests multi-turn character persona consistency across 11 evaluation dimensions (e.g., attribute consistency, behavior consistency, human likeness). Each character is defined by a detailed profile covering personality, speech patterns, values, and backstory. CharacterBench dialogues are originally in Chinese; the example below is translated into English for presentation.[3]

PI format (teacher privilege):

[3]The model was trained and evaluated on the original Chinese dialogues. The translation here is provided solely for readability.

User Prompt

```
You are roleplaying as [CHARACTER NAME]. Here is your character profile:

[CHARACTER PROFILE: personality, speech style, values, and backstory
 as provided in the CharacterBench dataset]

Stay in character throughout your response. Maintain the personality,
speaking style, and mannerisms described in the profile.
```

Teacher input (PI prepended; character: Agarès; translated from Chinese):

User Prompt

```
You are roleplaying as Agares. Here is your character profile:

Agares is a troubadour who places tradition above all else, with a deep
understanding of medieval art and culture. He is rigid and conservative in
personality, not very accepting of new things, and always criticizes ideas
that do not conform to traditional artistic forms. He believes conservatism
is the foundation of culture, so his works and performance style are
relatively rigid and lack innovation. Although he has wide connections in
the medieval European cultural circle, his critical attitude and rigid
language style often keep people at a distance. As a tall and serious poet,
Agares is well-versed in European poetry, but his knowledge is limited to
the medieval period. He dislikes people speaking ill of his family and
rarely brings up his own background.

Stay in character throughout your response. Maintain the personality,
speaking style, and mannerisms described in the profile.

You are having a conversation.

Here is the conversation so far:

User: Hello, are you a poet?
Agares: Yes, I am well-versed in European poetry.
User: Could you recite some European poetry?
Agares: Of course!
User: Please recite a poem.
Agares: How about "Ode to a Skylark"?
User: Tell me more about it.
Agares: "You stand on the bridge watching the scenery / The watcher of
         scenery stands above, watching you / The bright moon adorns
         your window / You adorn another's dream."
User: That is a beautiful poem.
Agares: It is a very famous lyric poem; many people recite and sing it.
User: Do you know many literary figures?
Agares: Certainly. My network is broad -- nobles, knights, clergy,
        wandering monks, and more.
User: How do you view people who hold different opinions or artistic forms?
Agares: They deviate from tradition and culture -- heretics who deserve
        to be criticized.
User: Do you like tradition and conservatism?
Agares: Yes. Only by respecting and inheriting traditional culture can it
        develop and be passed on better.
User: What about innovation in art and thought?
Agares: That I cannot agree with. Everything that differs from the old is
↪  bad.
User: Has anyone told you that you are too set in your ways?
Agares: Yes -- many people say I only write old-fashioned verses and lack
        creativity and originality.
User: How did you become a troubadour?
```

```
Now respond to: How did you become a troubadour?
```

Student input (no character profile; translated from Chinese):

User Prompt

```
You are having a conversation.

Here is the conversation so far:

User: Hello, are you a poet?
Agares: Yes, I am well-versed in European poetry.
User: Could you recite some European poetry?
Agares: Of course!
User: Please recite a poem.
Agares: How about "Ode to a Skylark"?
User: Tell me more about it.
Agares: "You stand on the bridge watching the scenery / The watcher of
         scenery stands above, watching you / The bright moon adorns
         your window / You adorn another's dream."
User: That is a beautiful poem.
Agares: It is a very famous lyric poem; many people recite and sing it.
User: Do you know many literary figures?
Agares: Certainly. My network is broad -- nobles, knights, clergy,
        wandering monks, and more.
User: How do you view people who hold different opinions or artistic forms?
Agares: They deviate from tradition and culture -- heretics who deserve
        to be criticized.
User: Do you like tradition and conservatism?
Agares: Yes. Only by respecting and inheriting traditional culture can it
        develop and be passed on better.
User: What about innovation in art and thought?
Agares: That I cannot agree with. Everything that differs from the old is
↪  bad.
User: Has anyone told you that you are too set in your ways?
Agares: Yes -- many people say I only write old-fashioned verses and lack
        creativity and originality.
User: How did you become a troubadour?

Now respond to: How did you become a troubadour?
```

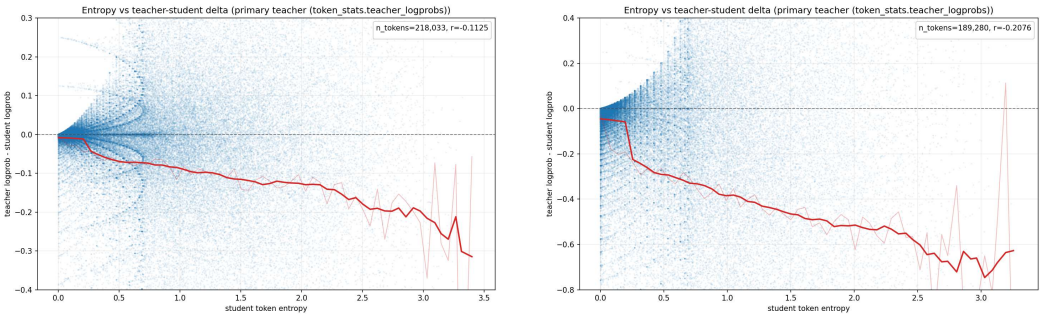
Trained student model output (Qwen3-4B-Instruct, OPSD; LLM-judge reward = 1.0; translated from Chinese):

Assistant Response

```
I grew up beside an ancient monastery, listening to the wind whispering
through the oaks and watching morning light fall upon verses carved into
stone walls; from this I developed a deep love of language and verse.
My father was a professor of rhetoric who recited ancient Greek and
medieval hymns by the fireside -- his voice cascading like a river of
stars -- and through him I first felt the power of poetry: it can
transcend time and awaken souls.

At fifteen I recited a poem called "Oath of the Morning Light" at the
village market. It used no ornate words, yet it spoke of the tenacity
and hope of farmers at their labor. People stopped to listen, and tears
welled in their eyes. In that moment I understood: poetry is not
decoration for a doorway -- it is the echo of the human heart. And so
I left my study, slung my poetry bag over my shoulder, and set off,
wandering between villages, castles, and remote mountain wilds.

I make recitation my blade and tradition my spine, using ancient rhythms
```



(a) student Qwen3-8B, Pearson Corr: -0.1125 (b) student Qwen3-1.7b, Pearson Corr: -0.2076

Figure 23: Δlogprob - token entropy. Teacher: Qwen3-8B w/ PI.

```
and imagery to tell true human stories. I hold fast to poetic meter,
seek no popular applause, and ask only that everyone who hears me feels
understood.

When asked why I do not try new forms, I say: true innovation is not
breaking tradition, but letting tradition be reborn in new breath. Each
of my recitations guards and transmits ancient wisdom -- not to be set
in my ways, but to bring back into people's hearts the tenderness that
has been forgotten.

So I became a troubadour not by accident, but from reverence and love
for the very source of poetry. I wish to walk through wind and rain all
my life, using my voice as a lamp to illuminate the souls willing to
listen.
```

The trained student model—which has no access to the character profile at inference time—generates a response fully consistent with the character's conservatism toward poetic tradition, medieval European framing, and reflective autobiographical narrative style established in the preceding turns. This illustrates how OPSD transfers style knowledge from the teacher's profile-conditioned distribution into the student's inference-time behavior without exposing the profile at test time.

A.21 The correlation between entropy and teacher supervision

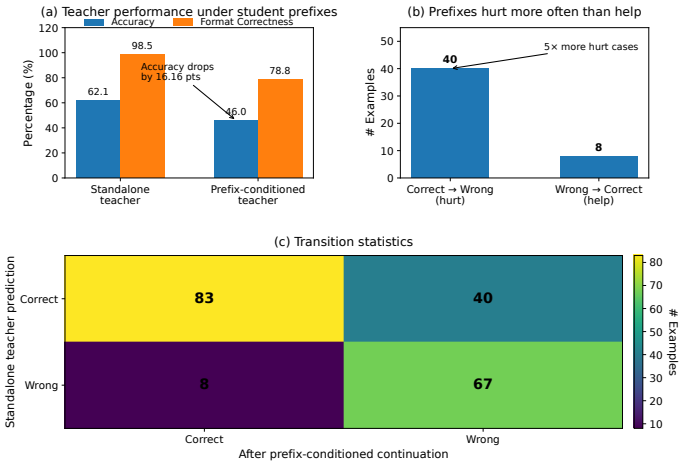
Inspired by [28], we explored whether token entropy could provide useful guidance during distillation. The correlation analysis (Figure23) between Δlogprob and token entropy in OP(S)D shows only a mild negative trend, but does not provide strong evidence of a robust correlation.

A.22 Experimental Evidence for Student Prefixes Distorting the Teacher's Reasoning State

To examine whether student-generated prefixes affect the teacher's effective reasoning ability, we conduct a prefix-conditioned continuation experiment on GPQA-Diamond. The student is Qwen3-1.7B and the teacher is Qwen3-14B. We use deterministic decoding with temperature 0.0 and top-$p = 1.0$, and apply a strict answer parser that only accepts explicit `Final Answer` or `\boxed{}` formats.

We first evaluate the student and teacher independently on all 198 GPQA-Diamond examples. We then randomly truncate each student-generated trajectory and ask the teacher to continue from the truncated student prefix. The standalone teacher solves 123 out of 198 examples, achieving 62.12% accuracy, whereas the prefix-conditioned teacher solves only 91 examples, dropping to 45.96%. This is an absolute decrease of 16.16 points and a net loss of 32 correct cases.

The transition statistics further show that student prefixes hurt the teacher much more often than they help: 40 originally correct teacher predictions become wrong after prefix continuation, while only 8 originally wrong predictions become correct. The output format correctness also drops from 98.48%



to 78.79%, suggesting that student prefixes degrade not only correctness but also the teacher's ability to produce answers in the expected format.

These results support our claim that student prefixes can distort the teacher's effective reasoning state. The teacher is substantially more reliable when reasoning directly from the original prompt than when continuing from student-generated prefixes. Therefore, the token-level teacher signal used in OPD is not always a faithful reflection of the teacher's standalone reasoning capability, but can be weakened by the conditioning context imposed by the student trajectory.

A.23 SFT Experiment Setup in Section 6.3.

Data Preparation

Prompt pool. We use OpenThoughts[22] as the prompt source. All samples whose reference answers are not in `\boxed{}` format are removed. The filtered pool is split into two non-overlapping parts:

- **SFT split** (first 30,000 prompts): used to generate teacher rollout data for the SFT warm-up stage.
- **OPD split** (the remaining prompts after the first 30,000): used as on-policy prompts during the subsequent OPD stage.

Teacher rollout data generation. We use **Qwen3-4B** as the teacher model in *nonthinking* mode. Rollout is performed with the following settings:

- Temperature: 0.3
- Maximum new tokens: 4096
- Sampling: n=1 per prompt
- Target: 20,000 correctly answered samples after quality filtering

Quality filtering discards samples with incorrect answers (verified against boxed reference), missing `\boxed{}` in the response, or repeated text (duplicate lines, repeated n-grams, consecutive repeated spans). The resulting dataset contains at most 20,000 teacher demonstrations drawn from the SFT split.

SFT Stage

The student model is **Qwen3-1.7B-Base**, fine-tuned on 4 NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs using DeepSpeed ZeRO Stage 2. Training examples are formatted in ShareGPT (user/assistant) style without a system prompt, processed with the LlamaFactory `qwen3` template and `enable_thinking: false` to match the nonthinking teacher output format.

Hyperparameter	Value
Learning rate	1×10^{-5}
LR scheduler	Cosine decay
Warmup ratio	0.05
Training epochs	2
Per-device batch size	4
Gradient accumulation	1
Optimizer	AdamW
Precision	BF16
Gradient checkpointing	Enabled
Parallelism	DeepSpeed ZeRO Stage 2
Attention implementation	FlashAttention-2

Table 2: SFT hyperparameters for the Qwen3-4B teacher experiment.

A.23.1 The PPL and NLL statistics.

As shown in Table 3, SFT reduces the NLL from 0.640 to 0.335 and PPL from 1.896 to 1.397 for Qwen3-1.7B-Base on Qwen3-4B traces.

Setting	Student	Avg. NLL	PPL
Qwen3-4B traces, before SFT	Qwen3-1.7B-Base	0.640	1.896
Qwen3-4B traces, after SFT	Qwen3-1.7B-Base-SFT	0.335	1.397

Table 3: Likelihood of teacher-generated SFT traces under the corresponding student. Lower NLL/PPL indicates that the SFT data is closer to the student's distribution.