DOES ON-POLICY DISTILLATION REALLY DISTILL? FROM NOISY TEACHER TO SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) offers dense token-level supervision as an alternative to the sparse outcome-level advantages of reinforcement learning with verifiable rewards (RLVR). However, the teacher scores student-generated trajectories that are inherently off-policy for it, so the reliability of its supervision, and hence the source of the student's improvement, remains unclear. We quantitatively analyze teacher supervision during OPD training and find substantial noise whose prevalence increases with teacher scale. Surprisingly, the student policy is insensitive to such noise, converging to comparable performance regardless of whether noisy supervision is retained or removed. Does OPD distill at all? By analyzing what drives its gains, we find that learning concentrates on low log-probability tokens, and using a single fixed negative advantage matches the performance of teacher-provided ones. This suggests that OPD works largely by suppressing low log-probability tokens, which requires no teacher. These findings motivate On-Policy Self-Adaptation (OPSA), a supervision-free method using entropy-adaptive negative advantages. It assigns stronger learning signals to high-entropy positions, suppressing tail tokens, and evenly redistributing probability mass among head tokens. Compared with the base `Qwen3-1.7B`, OPSA improves Avg@32 by 35.41 points on AIME24, corresponding to a 263% relative gain, and more than doubles Pass@32 across all three benchmarks. It also outperforms OPD by 16.77 points in Avg@32 on AIME24. Extensive experiments and analyses across model families and tasks further demonstrate its effectiveness and generalizability.

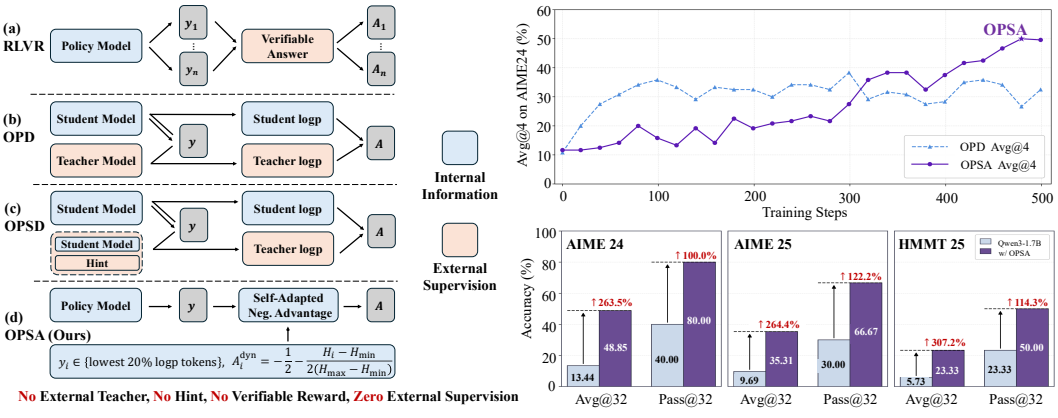


Figure 1: **Left:** Overview of different on-policy reinforcement learning algorithms. Unlike existing methods that derive advantages from external supervision, OPSA enables self-improvement with no external supervision by assigning entropy-adaptive negative advantages to low-probability tokens. This suppresses tail tokens and redistributes their mass among head tokens, sharpening low-entropy positions while preserving diversity at high-entropy forks. **Right:** Training dynamics and performance of OPSA. OPSA eliminates the teacher supervision used in OPD, and further outperforms OPD by 12% in Avg@4. On `Qwen3-1.7B`, OPSA improves Avg@32 by 263-307% across mathematical reasoning benchmarks and more than doubles Pass@32.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Reinforcement learning (RL) has become a dominant post-training paradigm to improve reasoning capabilities of large language models (LLMs) (Guo et al., 2025; Yang et al., 2025), offering stronger performance and generalization than offline methods such as supervised fine-tuning (SFT) (Chu et al., 2025; Chen et al., 2025). Reinforcement learning with verifiable rewards (RLVR), e.g., GRPO (Shao et al., 2024), samples multiple responses per question and assigns advantages by normalizing verifiable rewards within each group (Fig. 1(a)). However, response-level rewards provide coarse and sparse supervision for long-horizon reasoning (Yue et al., 2025). Moreover, when responses within a group share the same correctness outcome, their normalized advantages vanish (Ding et al., 2026), weakening the learning signal and making training unstable.

On-Policy Distillation (OPD) (Agarwal et al., 2024; Gu et al., 2024) addresses this limitation by using a strong teacher to provide token-level advantages through reverse Kullback-Leibler (KL) optimization on student-sampled trajectories (Fig. 1(b)) (Burnham & Anderson, 2001; Lu & Lab, 2025). However, it requires shared vocabularies and white-box access to teacher logits, restricting teacher selection. On-Policy Self-Distillation (OPSD) (Hübotter et al., 2026; Zhao et al., 2026; Shenfeld et al., 2026) retains the same OPD paradigm but replaces the external teacher with the policy itself conditioned on hints, such as reference answers (Fig. 1(c)). Constructing such hints still requires additional sampling or annotation and may risk information leakage (Yang et al., 2026).

More fundamentally, OPD and OPSD share the same advantage-assignment paradigm, which asks the teacher to score student-sampled prefixes that are inherently off-policy for it. This raises the question of whether the teacher can still provide reliable supervision in such off-policy contexts (Fu et al., 2026; Hou et al., 2026). Our analysis reveals a substantial fraction of noisy teacher supervision, where *noisy* means a negative advantage on a correct answer or a positive advantage on an incorrect one, and this fraction grows with teacher scale. Surprisingly, training solely on these noisy trajectories achieves performance similar to both standard OPD and OPD trained with noisy trajectories removed, suggesting that OPD's gains may not arise from the behavior matching it is designed to achieve. This raises a fundamental question: *what actually drives student improvement in OPD?*

To answer this question, we analyze OPD's gains from the perspectives of token selection and learning signals. We find that (i) improvement is driven primarily by low log-probability tokens sampled by the student, and (ii) negative advantages are critical, as replacing all OPD advantages with a single fixed negative value yields comparable performance. These observations suggest that OPD's gains come from suppressing low-probability tokens sampled by the student, requiring no teacher. Fixed negative advantages reproduce much of OPD's gains, but they discard its fine-grained token-level signals by treating all tokens equally. We therefore study how much negative signal each token should receive, and find that token entropy determines it, with stronger signal at high-entropy positions giving better performance.

Altogether, we propose On-Policy Self-Adaptation (OPSA), a supervision-free token-level RL method that assigns entropy-adaptive negative advantages to low-probability tokens (Fig. 1(d)). We summarize our contributions as follows:

- We find that teacher supervision in OPD is highly noisy and uncover a surprising insensitivity of the student policy to such noise. Training exclusively with noisy supervision, excluding noisy supervision, and standard OPD all converge to comparable accuracy after similar numbers of training steps.
- We identify the key drivers of OPD improvement: effective learning is concentrated on low-logp student-sampled tokens, and fixed negative advantages on these tokens can match standard OPD. This suggests that OPD's gains stem less from teacher distillation than from suppressing low-probability tokens, questioning the necessity of teacher supervision.
- We propose On-Policy Self-Adaptation, a supervision-free token-level RL method. It reshapes the policy distribution by suppressing tail-token probabilities and redistributing the probability mass among head tokens. This sharpens the distribution at low-entropy positions while preserving diversity at high-entropy fork tokens to support effective exploration.
- We show through extensive experiments that OPSA generalizes across model families and tasks. On `Qwen3-1.7B`, OPSA improves Avg@32 by 263%–307% relative to the base model across the three benchmarks and more than doubles Pass@32 on each benchmark.

2 TEACHER SUPERVISION IN OPD IS HIGHLY NOISY, BUT STUDENTS IMPROVE REGARDLESS

2.1 PRELIMINARY

On-Policy Distillation (OPD) (Agarwal et al., 2024; Gu et al., 2024) optimizes the reverse Kullback–Leibler (KL) divergence between the student and teacher distributions over the same prefixes, which is computed using the K1 estimator (Lu & Lab, 2025):

$$\mathrm{KL}(\pi_s \,||\, \pi_t) = \mathbb{E}_{y\sim\pi_s(\cdot|x)} \sum_{i=1}^{|y|} \left[\log \pi_s(y_i \mid x; y_{<i}) - \log \pi_t(y_i \mid x; y_{<i})\right], \tag{1}$$

where $y_{<i} = [y_1, \cdots, y_{i-1}]$ denotes prefixes of the response y sampled from student policy π_s. In practice, OPD provides token-level advantage signals A_i for the student policy during the Reinforcement Learning (RL) training, and the loss function can be written as:

$$\mathcal{L}_{\text{OPD}} = -\mathbb{E}\left[\frac{1}{|y|}\sum_{i=1}^{|y|} A_i \log \pi_s(y_i|x; y_{<i})\right], \quad A_i = \mathrm{sg}\left[\log \frac{\pi_t(y_i \mid x; y_{<i})}{\pi_s(y_i \mid x; y_{<i})}\right]. \tag{2}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation. The loss assigns teacher-derived positive or negative credit to student sampled tokens, pushing the student toward the teacher at each state.

2.2 NOISY TEACHER SIGNALS IN OPD

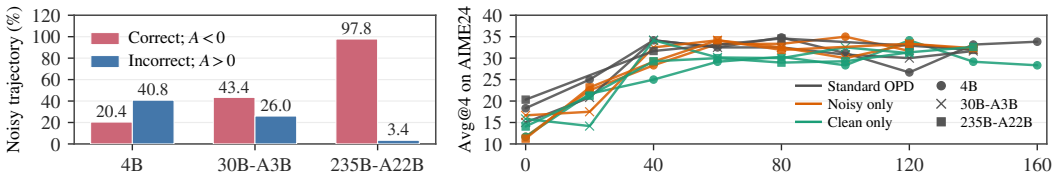
Since the KL divergence is computed conditioned on prefixes sampled from the student distribution, the teacher must score trajectories it would not generate. This raises a key concern: can the teacher provide reliable supervision in such off-policy settings (Fu et al., 2026; Xie et al., 2026)?

Definition of Noisy Signals and Setup. Previous work (Hou et al., 2026) often measures supervision noise at the trajectory level. However, the quality of intermediate reasoning steps is difficult to assess quantitatively, as a per-token reference for what the supervision should be is generally unavailable. Therefore, we focus on tokens corresponding to verifiable final answers enclosed in `\boxed{}`, whose correctness is determined by the verifier. We call the supervision *noisy* when the sign of the teacher-provided advantage on these tokens disagrees with the verifiable reward, i.e., when incorrect answers receive positive advantages or correct answers receive negative ones. Such supervision provides the student with misleading learning signals during RL training. We use `Qwen3-1.7B` as the student policy π_s with thinking mode disabled, and use `Qwen3-4B/30B-A3B/235B-A22B-Instruct` as teacher π_t, respectively. For the following analysis, we randomly sample one correct and one incorrect response from π_s for each of 500 questions in DAPO-17k (Yu et al., 2025), with correctness verified based on ground-truth answers.

Teacher Supervision is Highly Noisy, and Noisier at Scale. As shown in Fig. 2(a), teacher supervision contains substantial noise. For the 4B teacher, 20.4% of correct trajectories receive negative advantages on their answer tokens, while 40.8% of incorrect trajectories receive positive advantages, yielding an overall noise rate of 30.6%. The noise rate increases with teacher size, from 30.6% for the 4B teacher to 34.7% for the 30B-A3B teacher and 50.6% for the 235B-A22B teacher. Strikingly, the largest teacher assigns negative advantages to 97.8% of correct and 96.6% of incorrect answer tokens, making its supervision largely insensitive to answer correctness. We attribute this to the growing distributional mismatch between the student and teacher policies, which makes student-generated trajectories increasingly off-policy from the teacher's perspective.

2.3 STUDENTS ARE INSENSITIVE TO NOISE IN TEACHER SUPERVISION

Section 2.2 reveals substantial noisy token-level supervision in OPD. To assess its effect, we partition trajectories by whether they contain noisy signals and compare standard OPD with variants trained only on noisy or non-noisy trajectories. As shown in Fig. 2(b), all three settings converge to comparable performance after a similar number of gradient steps. Remarkably, even when training is restricted to trajectories with noisy advantages, the student improves at a rate comparable to standard OPD. This finding reveals the insensitivity of OPD to supervision noise in the teacher-provided



(a) Reward-advantage direction mismatch (b) Training dynamics under different supervision

Figure 2: Performance of `Qwen3-1.7B` under teachers of different scales. (a): The proportion of teacher-provided advantage signals whose directions disagree with verifiable rewards on student-sampled trajectories. Red and blue bars denote noise rates in correct and incorrect trajectories, respectively. (b): AIME24 Avg@4 during OPD training with different advantage-signal variants.

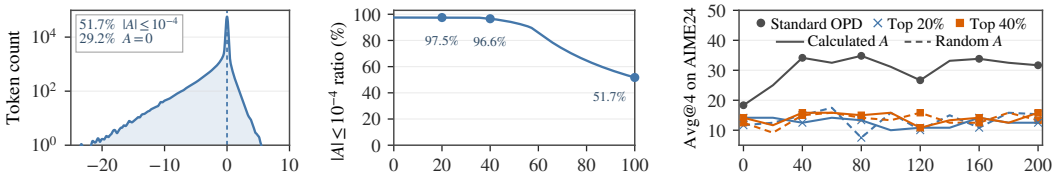
advantages. In particular, when the supervision is substantially noisy, the student's improvement may not come from matching the teacher's behavior. This raises a more fundamental question:

Research Question

OPD improves student performance even when teacher supervision is highly noisy. This suggests that the gains may not come from knowledge transfer, the mechanism OPD is built on. What, then, drives student improvement in OPD?

3 Where Does Student Improvement Come From?

In this section, we disentangle OPD's gains by identifying which tokens drive improvement (Section 3.1) and which learning signals are effective (Section 3.2).



(a) Advantage distribution (b) Zero advantage dominates high logp (c) Training across top-logp fractions

Figure 3: `Qwen3-1.7B` with a 4B-Instruct teacher. (a) OPD token-level advantage distribution. (b): Fraction of near-zero advantages among tokens within different top-percentile ranges of student logp. (c) Training on top-logp tokens yields little improvement, indicating weak learning signals.

3.1 Which Tokens Contribute to OPD Training?

Revisiting the loss in Eq. 2, let z_t^v denote the logit for token v conditioned on context $y_{<t}$. We have the logit-level gradients for each token (Zhu et al., 2025a; Jia et al., 2026):

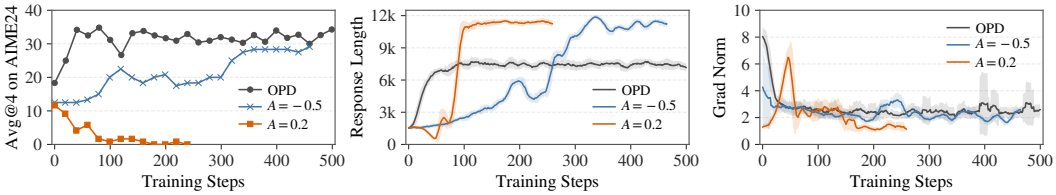
$$-\frac{\partial \mathcal{L}_{\text{OPD}}}{\partial z_t^v} \propto \begin{cases} A_t \left(1 - \pi_s(v \mid x; y_{<t})\right), & \text{if } v = y_t \text{ (sampled token)} \\ -A_t(\pi_s(v \mid x; y_{<t})), & \text{if } v \neq y_t \text{ (unsampled token)} \end{cases}. \tag{3}$$

We observe that the gradient vanishes in two regimes: when $|A_t|$ is small, and when $\pi_s(y_t \mid x, y_{<t})$ approaches one for the sampled token. Tokens in either regime have negligible impact on training.

Near-Zero Advantages Overlap with High-Logp Student Tokens. As shown in Fig. 3(a), OPD produces a large mass of near-zero token-level advantages: 29.2% of tokens have exactly 0 advantage, and 51.7% satisfy $|A| < 10^{-4}$. Fig. 3(b) reveals a substantial overlap between these near-zero-advantage tokens and the student's high-logp tokens. At such positions, the student and teacher assign similarly high probabilities under the same prefix, yielding only weak advantage signals.

High-Logp Tokens Contribute Nothing to On-Policy Training. To test whether high-logp tokens provide effective learning signals, we train on varying proportions of the student's top-logp tokens using either OPD-computed or random advantages sampled uniformly from $[-1, 1]$. As

shown in Fig. 3(c), neither setting noticeably improves AIME24 performance. Interestingly, the performance remains essentially unchanged even when the original advantages are replaced with random values. These results suggest that high-logp tokens provide no effective learning signal during on-policy training, largely regardless of the advantages assigned to them.



(a) Avg@4 on AIME24 (b) Response Length (c) Grad Norm

Figure 4: Training dynamics of standard OPD and on-policy training with fixed advantages.

3.2 Which Learning Signals Drive Policy Improvement?

Since OPD computes reverse KL on student-sampled trajectories that are naturally off-policy for the teacher, many sampled tokens receive lower teacher probabilities and thus negative advantages (Fig. 3(a)). Interestingly, negative signals alone have also been shown to improve policy performance in RL, as demonstrated by Negative Sample Reinforcement (Zhu et al., 2025a). This motivates us to ask whether OPD's gains primarily arise from its dominant negative advantages rather than teacher-specific supervision. We use `Qwen3-1.7B` as the student and compare standard OPD with two teacher-free variants using fixed advantages of -0.5 and $+0.2$. Since high-logp tokens provide little effective gradient (Section 3.1), training is restricted to the lowest-20%-logp tokens.

A Fixed Negative Advantage Alone Improves Students. Fig. 4 shows that restricting OPD to the lowest-20%-logp tokens matches full-token OPD, supporting Section 3.1. Additionally, a fixed negative advantage steadily improves AIME24 Avg@4 and produces response-length growth similar to OPD, stabilizing around 12K tokens. In contrast, a fixed positive advantage causes rapid policy collapse, with response length approaching zero, exploding gradient norms, and degenerate outputs.

Takeaway

Our analysis suggests that OPD gains may arise not from distilling teacher knowledge, but from suppressing low-probability tokens sampled by the student itself, an operation that requires no teacher at all. This observation motivates the method introduced in Section 4.

4 Methodology

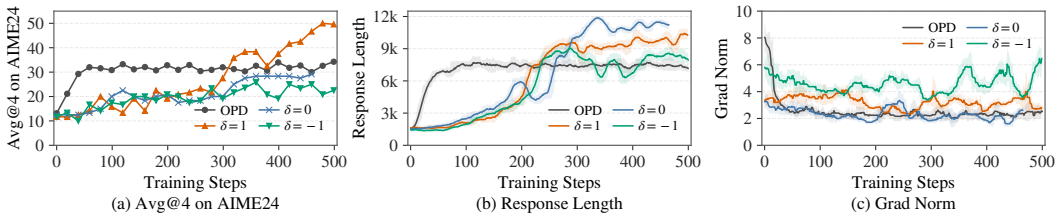
Section 3 showed that OPD's improvements survive the removal of the teacher, of positive advantages, and of all but the lowest-logp tokens. This leaves one key question: how should negative signals be assigned across low-logp tokens? Section 4.1 identifies token entropy as the key factor, Section 4.2 introduces On-Policy Self-Adaptation (OPSA), and Section 4.3 analyzes its mechanism.

4.1 Entropy Determines the Amount of Negative Signals

Section 3.1 shows that low-logp tokens drive learning, but low logp can arise from either high uncertainty, where probability is spread across many plausible tokens, or a confident distribution in which a rare tail token is sampled. Wang et al. (2025) have shown that high-entropy tokens play an important role in RL and may benefit from stronger learning signals. Motivated by this, we compare two entropy-aware schemes that dynamically adjust the magnitude of the negative advantage:

$$A_i^{\text{dyn}} = A_i^{\text{fix}} - \frac{1}{4}\delta \cdot r_i, \quad r_i = 2\frac{H_i - H_{\min}}{H_{\max} - H_{\min}} - 1, \tag{4}$$

where $H_{\min}$ and $H_{\max}$ denote the minimum and maximum entropy among the lowest-20%-logp positions in each response, yielding a normalized entropy score $r_i \in [-1, 1]$. The parameter δ controls how advantage magnitude varies with entropy: $\delta = 1$ emphasizes high-entropy tokens, $\delta = -1$ emphasizes low-entropy tokens, and $\delta = 0$ recovers the fixed-negative setting in Section 3.2.



(a) Avg@4 on AIME24 (b) Response Length (c) Grad Norm

Figure 5: Training dynamics of on-policy training with dynamic negative advantages.

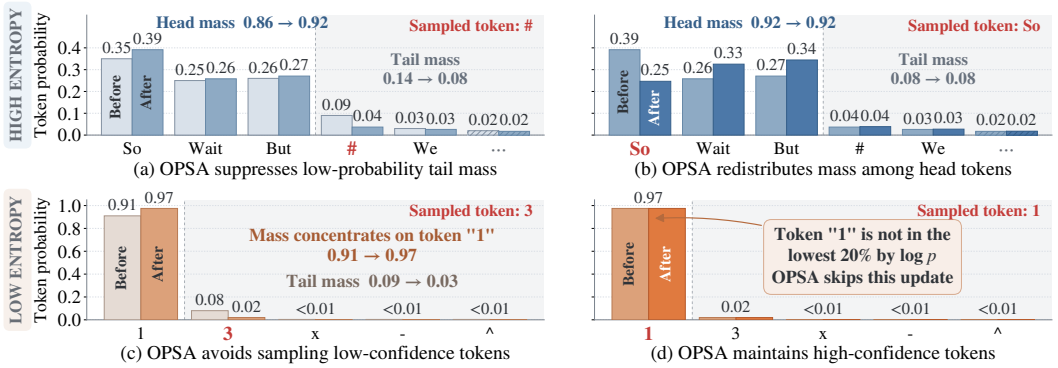
Adaptive Advantages with $\delta = 1$ Outperform OPD. The results in Fig. 5 show that emphasizing high-entropy tokens ($\delta = 1$) yields steady improvement, reaching 50.0% Avg@4 on AIME24 versus 35.13% for standard OPD. In contrast, the $\delta = -1$ variant is less stable, exhibits higher gradient norms, and underperforms the fixed-negative baseline. These results highlight the importance of assigning stronger negative learning signals to high-entropy tokens.

4.2 On-Policy Self-Adaptation

These results give a complete on-policy training recipe, which we call **O**n-**P**olicy **S**elf-**A**daptation (**OPSA**). OPSA updates only the lowest-logp tokens, assigns them negative advantages, and scales the magnitude with entropy. It requires no external supervision of any kind, including teacher signals, ground-truth rewards, or reference answers. Setting $A_i^{\text{fix}} = -\frac{3}{4}$ and $\delta = 1$ in Eq. (4) and restricting the update to $\mathcal{S}_{\text{lowest20}}$, the training objective of OPSA is defined as follows:

$$\mathcal{L}_{\text{OPSA}} = -\mathbb{E}\left[\frac{1}{|\mathcal{S}_{\text{lowest20}}|}\sum_{i\in\mathcal{S}_{\text{lowest20}}} A_i^{\text{dyn}} \log \pi_\theta(y_i|x; y_{<i})\right], \quad A_i^{\text{dyn}} = \text{sg}\left[-\frac{1}{2} - \frac{H_i - H_{\min}}{2(H_{\max} - H_{\min})}\right], \tag{5}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation. OPSA eliminates teacher forward passes by deriving advantages directly from the student's token-level entropy, introducing negligible overhead and substantially reducing training time. Detailed efficiency results are provided in Appendix B.4.



(a) OPSA suppresses low-probability tail mass (b) OPSA redistributes mass among head tokens

(c) OPSA avoids sampling low-confidence tokens (d) OPSA maintains high-confidence tokens

Figure 6: Examples of OPSA updates. (a) At high-entropy positions, OPSA suppresses sampled tail tokens and reallocates probability to head tokens. (b) At high-entropy positions, OPSA redistributes probability among head tokens, with little effect on the tail. (c) Avoiding tail-token sampling at low-entropy positions. (d) Preserving high-confidence predictions.

4.3 Why Does OPSA Work?

To understand OPSA's mechanism, we categorize tokens by entropy and sampled-token probability into four cases: high-/low-entropy head and tail tokens. Based on the logit-level gradients in Eq. 3, Fig. 6 illustrates how OPSA's negative advantages reshape the policy distribution in each case.

Tail Tokens: OPSA Suppresses Low-Probability Reasoning Branches. Stochastic rollouts may occasionally sample low-probability tail tokens, steering reasoning into unlikely branches. By assigning negative advantages to the lowest-20%-logp tokens, OPSA suppresses these sampled tail tokens and reallocates their probability mass toward higher-probability alternatives (Fig. 6(a,c)), thereby steering reasoning away from low-probability branches.

Table 1: Performance of OPSA across models on in-domain math and out-of-domain code and Q&A tasks. Qwen models use non-thinking mode; Olmo IT and TK denote Instruct and Think variants.

Methods	Math (I.D.)						O.O.D.	
	AIME24		AIME25		HMMT25		MBPP+	GPQA_D
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	
Qwen3-1.7B	13.44	40.00	9.69	30.00	5.73	23.33	58.24	27.92
w/ OPSA	48.85	80.00	35.31	66.67	23.33	50.00	59.44	32.40
Δ	+35.41	+40.00	+25.62	+36.67	+17.60	+26.67	+1.20	+4.48
	↑ 263.5%	↑ 100.0%	↑ 264.4%	↑ 122.2%	↑ 307.2%	↑ 114.3%	↑ 2.1%	↑ 16.0%
Qwen3-4B	23.33	56.67	20.52	56.67	13.13	33.33	66.93	38.46
w/ OPSA	62.08	83.33	58.44	83.33	37.40	60.00	68.35	41.29
Δ	+38.75	+26.66	+37.92	+26.66	+24.27	+26.67	+1.42	+2.83
	↑ 166.1%	↑ 47.0%	↑ 184.8%	↑ 47.0%	↑ 184.8%	↑ 80.0%	↑ 2.1%	↑ 7.4%
Qwen3.5-9B	76.35	93.33	56.04	93.33	44.48	86.67	77.33	70.53
w/ OPSA	87.81	96.67	76.98	96.67	67.40	93.33	79.27	73.70
Δ	+11.46	+3.34	+20.94	+3.34	+22.92	+6.66	+1.94	+3.17
	↑ 15.0%	↑ 3.6%	↑ 37.4%	↑ 3.6%	↑ 51.5%	↑ 7.7%	↑ 2.5%	↑ 4.5%
$\text{Olmo3-7B}_{\text{IT}}$	54.48	83.33	38.75	76.67	24.90	56.67	57.79	34.12
w/ OPSA	61.88	93.33	47.92	83.33	29.69	63.33	59.61	35.64
Δ	+7.40	+10.00	+9.17	+6.67	+4.79	+6.66	+1.82	+1.52
	↑ 13.6%	↑ 12.0%	↑ 23.7%	↑ 8.7%	↑ 19.2%	↑ 11.8%	↑ 3.1%	↑ 4.4%
$\text{Olmo3-7B}_{\text{TK}}$	71.88	93.33	61.77	90.00	41.98	73.33	64.55	38.86
w/ OPSA	77.29	93.33	67.08	93.33	47.50	83.33	66.33	40.93
Δ	+5.41	+0.00	+5.31	+3.33	+5.52	+10.00	+1.78	+2.07
	↑ 7.5%	↑ 0.0%	↑ 8.6%	↑ 3.7%	↑ 13.1%	↑ 13.6%	↑ 2.8%	↑ 5.3%

Head Tokens: OPSA Evenly Redistributes Mass Without Collapsing Diversity. As tail tokens are suppressed, sampling becomes increasingly head-dominated. At high-entropy positions, however, OPSA does not collapse onto a single prediction: a negative advantage on a sampled head token redistributes probability mass among competing head tokens (Fig. 6(b)), preserving diverse reasoning branches. At low-entropy positions, high-confidence tokens are typically excluded from the lowest-20%-logp set, preserving precise predictions (Fig. 6(d)) (Zhang et al., 2026).

Reshaped Distribution Enables Reflective Reasoning. High-entropy positions are closely associated with reflective reasoning (Ding et al., 2026; Zhao et al., 2025a), where tokens such as "wait" and "but" often initiate reflection or self-correction. By increasing the total probability mass of head tokens while redistributing it across competing alternatives, OPSA makes such reflective branches more likely to be explored. Appendix C.1 and C.9 provide more quantitative analysis.

5 Experiments

5.1 Experimental Setup

Models and Datasets. We evaluate OPSA on models from the Qwen3 (Yang et al., 2025), Qwen3.5 (Team, 2026), and Olmo3 (Olmo et al., 2025) families. Models are trained on DAPO-17k (Yu et al., 2025) using only questions, without labels or ground-truth answers. We evaluate on AIME24, AIME25, and HMMT25 for in-domain mathematical reasoning, and MBPP+ (Liu et al., 2023) and GPQA-Diamond (Rein et al., 2023) for out-of-domain generalization. We provide additional model details and code-task training results in Appendix B.2 and C.3, respectively.

Baselines and Implementation. We compare OPSA with GRPO (Shao et al., 2024), TTRL (Zuo et al., 2025), OPD (Lu & Lab, 2025), and OPSD (Zhao et al., 2026). OPSA is implemented with `slime` (Zhu et al., 2025b), optimizing Eq. 5 over the lowest-20%-logp sampled tokens. Full model, training, evaluation, and baseline details are provided in Appendix B.2.

5.2 Main Results and Discussions

OPSA Consistently Improves Different Models. Table 1 shows that OPSA consistently improves performance across model families, scales, and reasoning modes. On the Qwen3 series, OPSA more than doubles Avg@32 across all mathematical reasoning benchmarks; for

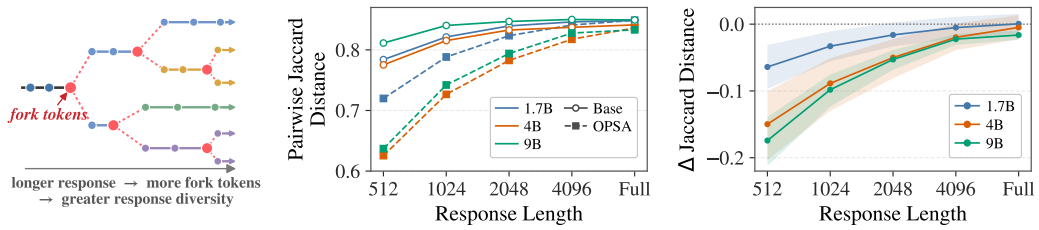
Table 2: Comparison of different on-policy RL methods on `Qwen3-1.7B`.

Methods	AIME24		AIME25		HMMT25		Average	
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32
Qwen3-1.7B	13.44	40.00	9.69	30.00	5.73	23.33	9.62	31.11
+ GRPO	33.96	70.00	25.31	50.00	15.10	43.33	24.79	54.44
+ TTRL	19.90	30.00	9.79	30.00	5.73	23.33	11.81	27.78
+ OPD	32.08	73.33	20.52	50.00	13.85	40.00	22.15	54.44
+ OPSD	33.33	73.33	22.50	53.33	14.90	43.33	23.58	56.67
+ OPSA	48.85	80.00	35.31	66.67	23.33	50.00	35.83	65.56
Δ=OPSA-RL$_{\text{best}}$	+14.89	+6.67	+10.00	+13.34	+8.23	+6.67	+11.04	+8.89
Qwen3-1.7B$_{\text{Thinking}}$	46.56	80.00	36.25	73.33	22.92	60.00	35.24	71.11
+ OPSA	52.50	83.33	42.79	73.33	26.85	60.00	40.71	72.22

`Qwen3-1.7B`, the relative gains reach 263% on AIME24/AIME25 and 307% on HMMT25. Improvements remain substantial on the stronger `Qwen3.5-9B`, with Avg@32 gains of 11.46 points on AIME24 and 22.92 points on HMMT25. The two `Olmo3-7B` variants show the same consistent trend despite their different baseline capabilities. OPSA also improves Pass@32 across nearly all settings. Even for strong models with near-saturated Pass@32, such as `Qwen3.5-9B` and `Olmo3-7B`$_{\text{TK}}$, OPSA maintains or further improves performance, including gains of 6.66 and 10.00 points on HMMT25, respectively. These results demonstrate that OPSA remains effective across models with substantially different initial capabilities.

Generalization of OPSA. To evaluate out-of-domain generalization, we test OPSA on MBPP+ and GPQA-Diamond. As shown in Table 1, OPSA consistently improves performance across models, demonstrating transfer beyond the training domain. For Qwen models, OPSA is trained with thinking mode disabled. Interestingly, the OPSA-trained model in non-thinking mode achieves performance comparable to the base model with thinking mode enabled, while enabling thinking mode after OPSA training yields further gains. This suggests that OPSA promotes longer-form reasoning behaviors while remaining complementary to the model's native thinking capability.

OPSA Outperforms Baselines Using Zero External Supervision. We compare OPSA with representative on-policy RL baselines. TTRL performs test-time training on AIME24 using its best checkpoint, while OPD uses `Qwen3-4B-Instruct` as the teacher. As shown in Table 2, OPSA consistently outperforms all baselines in both Avg@32 and Pass@32 without external supervision, exceeding the strongest baseline by 11.04 and 8.89 points on average, respectively. TTRL's self-consistency training tends to over-sharpen the policy distribution and degrade Pass@k. OPSD provides meaningful gains only when the student runs in non-thinking mode while the teacher runs in thinking mode, creating a large first-token distribution mismatch and strong negative advantages. OPD and GRPO achieve comparable overall performance, but OPD, OPSD, and OPSA require fewer rollout generations and are substantially more efficient than GRPO (Table 7, Appendix B.4).



(a) Tree-Style Reasoning Pattern (b) Pairwise 4-gram Jaccard Distance (c) Δ Jaccard Distance

Figure 7: (a) As reasoning length increases, more opportunities for branching arise. At each fork, OPSA distributes probability more evenly across alternative tokens, allowing multiple sampling to explore different reasoning paths and thereby increasing response diversity. (b–c) Jaccard-distance comparisons on AIME24 before and after OPSA training, using 32 sampled responses per problem.

OPSA Doesn't Hurt Models' Diversity. Although OPSA sharpens the overall token distribution, especially at low-entropy positions, it redistributes probability mass among competing head tokens at high-entropy forks. This mechanism allows OPSA to preserve response diversity in long-form

reasoning. We measure diversity using Jaccard distance (JD) (Broder, 1997; Wang & Wan, 2018). As shown in Fig. 7, the JD gap between the base and OPSA-trained models narrows with response length and eventually approaches zero. This result suggests that OPSA continues to explore alternative reasoning branches at successive fork positions, producing the branching, tree-like reasoning patterns illustrated in Fig. 7(a) without causing diversity collapse. The Pass@32 results in Table 1 further support that OPSA preserves effective exploration.

We provide more experiments and analyses in Appendix C, including OPSA training and extensions across different domains, ablation studies, and further investigation into its underlying mechanism.

6 Related Works

Supervision Signals in On-Policy Distillation. OPD optimizes a student policy on its self-generated trajectories. GKD (Agarwal et al., 2024) combines student-generated and supervised trajectories under flexible divergence objectives, while MiniLLM (Gu et al., 2024) adopts reverse-KL distillation to reduce exposure bias. More recently, Lu & Lab (2025) improve OPD efficiency for reasoning tasks using the K1 estimator. Another line of work (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026) replaces the external teacher with the same policy conditioned on additional hints, yielding on-policy self-distillation (OPSD). However, OPSD retains the OPD training paradigm, leaving unresolved whether the constructed teacher can provide reliable supervision on student-generated trajectories. Recent studies aim to avoid potential training noise in OPD through token selection (Xu et al., 2026; Fu et al., 2026) or fine-grained credit assignment (Xing et al., 2026; Hou et al., 2026), but still rely on external supervision. In contrast, we investigate the source of OPD's gains and propose OPSA, which enables self-improvement without external supervision.

Label-Free Policy Improvement. Label-free training enables a policy to improve using signals derived from the model itself, without external supervision. Self-rewarding methods (Yuan et al., 2024; Ding & Zhang, 2025) use an LLM-as-a-Judge (Zheng et al., 2023) to score the model's own responses and construct reward signals. TTRL (Zuo et al., 2025) and EMPO (Zhang et al., 2025) instead infer pseudo-gold answers from GRPO rollouts through majority voting or clustering, and then use them for reward assignment. Intuitor (Zhao et al., 2025b) adopts self-certainty, using trajectory-level entropy as a reward to encourage the policy toward more confident outputs. However, these methods depend heavily on the policy's initial capability. If the correct answer does not lie near a mode of the policy distribution, coarse trajectory-level rewards may reinforce an incorrect mode, causing the policy to become increasingly overconfident and ultimately leading to a collapse in Pass@k. In contrast, OPSA does not force the policy to converge toward any self-generated reward signal. Instead, it provides finer-grained token-level learning signals through entropy-adaptive negative advantages, simultaneously improving precision at low-entropy positions and exploration at high-entropy fork tokens, thereby enhancing the policy's overall reasoning capability.

7 Conclusion

The central contribution of this work is a systematic investigation into the source of OPD's performance gains. We find that much of its improvement can be reproduced without teacher supervision by assigning fixed negative advantages to low-probability tokens, while token-entropy-based scaling provides more effective fine-grained learning signals. Building on these findings, we introduce **O**n-**P**olicy **S**elf-**A**daptation (**OPSA**), an external-supervision-free framework for policy self-improvement. OPSA suppresses low-probability tail tokens and redistributes probability mass among competing head tokens, improving confidence at low-entropy positions while preserving exploration at high-entropy reasoning forks. This distribution reshaping promotes longer, more reflective reasoning and substantially improves both Avg@32 and Pass@32.

Notably, OPSA outperforms OPD without teacher supervision while exhibiting similar training dynamics, including response-length growth. This calls for a reexamination of OPD with the K1 estimator: its gains may arise not only from imitating a stronger teacher, but also from reshaping the student's own probability distribution. More broadly, our results show that the policy's internal token-level uncertainty can provide a fine-grained learning signal for on-policy self-improvement without rewards, reference answers, or teacher supervision.

ETHICS STATEMENT

This work studies post-training methods for improving the reasoning capabilities of language models. Our experiments use existing publicly available models and datasets and do not involve human subjects or personally identifiable information. While improved reasoning capabilities may have broader dual-use implications, our work does not target harmful or safety-critical applications.

REPRODUCIBILITY STATEMENT

We provide the full formulation of OPSA and its training objective in the main text. Detailed model configurations, training and evaluation hyperparameters, reasoning-mode settings, checkpoint selection procedures, computational resources, datasets, and benchmarks are documented in the Appendix B. After the review process, we plan to publicly release all training and evaluation code, together with the trained model checkpoints.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing. They were not used to generate experimental data or results. All AI-assisted content was carefully reviewed and verified by the authors, who take full responsibility for the final content of this work.

REFERENCES

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

Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pp. 21–29. IEEE, 1997.

Kenneth P Burnham and David R Anderson. Kullback-leibler information as a basis for strong inference in ecological studies. *Wildlife research*, 28(2):111–119, 2001.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Yi Ding and Ruqi Zhang. Sherlock: Self-correcting reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 38:101638–101672, 2025.

Yi Ding, Ziliang Qiu, Bolian Li, and Ruqi Zhang. Learning self-correction in vision–language models via rollout augmentation. In *Forty-third International Conference on Machine Learning*, 2026.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, et al. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026.

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

Nan Jia, Haojin Yang, Xing Ma, Jiesong Lian, Shuailiang Zhang, Weipeng Zhang, Ke Zeng, Xunliang Cai, and Zequn Sun. Asymmetric on-policy distillation: Bridging exploitation and imitation at the token level. *arXiv preprint arXiv:2605.06387*, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

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

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

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

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces. https://huggingface.co/datasets/open-r1/codeforces, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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

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

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Ke Wang and Xiaojun Wan. Sentigan: Generating sentimental texts via mixture adversarial networks. In *IJCAI*, pp. 4446–4452, 2018.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2025.

Yan Xie, Sijie Zhu, Tiansheng Wen, Bo Chen, and Yifei Wang. On the position bias of on-policy distillation. *arXiv preprint arXiv:2606.22600*, 2026.

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

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Tip: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026.

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

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 38:67345–67372, 2025.

Ruixiang Zhang, Richard He Bai, Huangjie Zheng, Navdeep Jaitly, Ronan Collobert, and Yizhe Zhang. Embarrassingly simple self-distillation improves code generation. *arXiv preprint arXiv:2604.01193*, 2026.

Jiachen Zhao, Yiyou Sun, Weiyan Shi, and Dawn Song. Can aha moments be fake? towards quantifying decorative and true thinking in chain-of-thought. *arXiv preprint arXiv:2510.24941*, 2025a.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *Advances in Neural Information Processing Systems*, 38:126546–126573, 2025a.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025b. GitHub repository. Corresponding author: Xin Lv.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2025.

A Limitations and Future Directions

Due to limited computational resources, our experiments focus on relatively small models of up to 9B parameters. It therefore remains unclear whether OPSA scales effectively to larger models or mixture-of-experts architectures. Moreover, our analysis suggests that OPSA primarily operates by redistributing the policy's existing probability mass. Its benefits may therefore be limited for heavily post-trained models whose output distributions are already overly sharp and exhibit very low entropy. Although OPSA requires no external annotations and can serve as a plug-and-play post-training method, it may not substantially expand the policy's underlying exploration frontier, as reflected by the relatively modest improvements in thinking-mode Pass@k. A promising direction is to combine OPSA's fine-grained negative-advantage assignment with other reinforcement learning methods to further improve policy exploration. In addition, OPSA outperforms OPD without teacher supervision while exhibiting similar training dynamics, including response-length growth. This finding calls for a reexamination of the mechanism underlying OPD with the K1 estimator and motivates further theoretical analysis to identify the true source of OPD's performance gains.

B Experimental Details

Table 3: Comparison of supervision signals in different on-policy training methods.

Methods	Dense Signal	No Verifiable Reward	No External Teacher	No Hint
RLVR (Shao et al., 2024)	✗	✗	✓	✓
TTRL (Zuo et al., 2025)	✗	✓	✓	✓
OPD (Lu & Lab, 2025)	✓	✓	✗	✓
OPSD (Zhao et al., 2026)	✓	✓	✓	✗
OPSA (Ours)	✓	✓	✓	✓

B.1 Comparison with Baselines

We compare OPSA with representative on-policy training methods, including reinforcement learning with verifiable rewards (RLVR), such as GRPO (Shao et al., 2024); Test-Time Reinforcement Learning (TTRL) (Zuo et al., 2025), which does not require external supervision; standard On-Policy Distillation (OPD) (Lu & Lab, 2025); and On-Policy Self-Distillation (OPSD) (Zhao et al., 2026), which replaces an external teacher with hints corresponding to questions. Table 3 summarizes the key properties of these methods. OPSA provides dense token-level learning signals while requiring neither verifiable rewards, external teacher models, nor auxiliary hints.

B.2 Training Configuration

Table 4: Training hyperparameters for OPSA.

Hyperparameter	Value
Framework	`slime` (`0.2.4`)
Training engine	Megatron (`0.16.0rc0`)
Rollout engine	SGLang (`0.5.14`)
GPUs	8xH100 GPUs
Learning rate	1×10^{-6}
Rollout batch size	64
n samples per prompt	1
Enable Thinking	`False`
Global batch size	64
Training decoding	Temperature 1.0, top-k -1, top-p 1.0, max response length 12000
Evaluation decoding	Temperature 0.7, top-k 20, top-p 0.8, max response length 32768
Save & Evaluation frequency	Save every 20 steps, validate every 20 steps
Checkpoint selection metric	Avg@4 on validation set

We provide the detailed OPSA training configuration in Table 4. For rollout during training, we adopt the default decoding parameters from the `slime` repository. For evaluation, we use the de-

Table 5: Detailed training setting for different models.

Models	Default Thinking Mode	Training Thinking Mode
`Qwen3-1.7B`	Hybrid	Non-Thinking
`Qwen3-4B`	Hybrid	Non-Thinking
`Qwen3.5-9B`	Hybrid	Non-Thinking
`Olmo3-7B-Instruct`	Non-Thinking	Non-Thinking
`Olmo3-7B-Think`	Thinking	Thinking

coding parameters recommended in the Qwen3 model card. Following prior work (Zhao et al., 2026; Hou et al., 2026), we select the best checkpoint for each method based on Avg@4 performance. To avoid ambiguity arising from different reasoning modes across models, Table 5 summarizes each model's default reasoning mode and the mode used for rollout generation during training.

B.3 Evaluation Configuration

Table 6: Evaluation hyperparameters for OPSA.

Hyperparameter	Value
Rollout engine	SGLang (`0.5.14`)
Temperature	0.7
Top-k	20
Top-p	0.8
Max new tokens	32768
Enable thinking	`False`
Samples per prompt	32
Metric	Avg@32 & Pass@32

We provide the detailed evaluation configuration in Table 6. We use `SGLang` for inference, generate 32 responses per prompt with thinking mode disabled, and report both Avg@32 and Pass@32.

B.4 More Details

Table 7: Training and inference overhead of OPSA compared to baselines.

Methods	Training Step Time (s)	Inference Token Counts	Inference Time (s)	Avg@32
Qwen3-1.7B	-	4457	1.78	13.44
+ GRPO	186.2	19108	5.27	33.96
+ OPD	61.2	15286	5.73	32.08
+ OPSA	46.3	23205	6.58	48.85
Qwen3-4B	-	8015	3.31	23.33
+ OPSA	68.6	20972	6.97	62.08
Qwen3.5-9B	-	6847	5.29	76.35
+ OPSA	214.7	9695	6.31	87.81

Training and Inference Overhead of OPSA. We compare the training and inference overhead of different methods in Table 7. OPSA and OPD train faster than GRPO because they do not require extensive rollout sampling to construct response groups. OPSA is even more efficient than OPD, as it neither deploys an additional teacher model nor performs forward passes through a larger and slower teacher. This allows more GPUs to be allocated to policy rollout and optimization. At inference time, OPSA steers the policy toward more reflective reasoning branches, producing longer responses while achieving higher accuracy.

Jaccard Distance in Fig 7. We measure response diversity using the pairwise Jaccard distance between sets of token-level 4-grams. For each response, we retain its first L tokens, where $L \in$

$\{512, 1024, 2048, 4096, \text{Full}\}$, and denote the resulting set of unique 4-grams by $\mathcal{G}_4(r^{(\le L)})$. The distance between two responses is defined as:

$$d_{\text{J}}(r_i, r_j; L) = 1 - \frac{\left|\mathcal{G}_4(r_i^{(\le L)}) \cap \mathcal{G}_4(r_j^{(\le L)})\right|}{\left|\mathcal{G}_4(r_i^{(\le L)}) \cup \mathcal{G}_4(r_j^{(\le L)})\right|}. \tag{6}$$

For each AIME24 problem, we average the distance over all response pairs among the $R = 32$ sampled responses, and then report the macro-average over all $P = 30$ problems:

$$D_{\text{J}}(L) = \frac{1}{P}\sum_{p=1}^{P} \frac{2}{R(R-1)} \sum_{1 \le i < j \le R} d_{\text{J}}(r_{p,i}, r_{p,j}; L). \tag{7}$$

A larger $D_{\text{J}}(L)$ indicates greater response diversity. We additionally report the difference between OPSA and the base model as

$$\Delta D_{\text{J}}(L) = D_{\text{J}}^{\text{OPSA}}(L) - D_{\text{J}}^{\text{Base}}(L). \tag{8}$$

C Additional Experimental Results and Analysis

C.1 OPSA Elicits Reflective Long-Form Reasoning

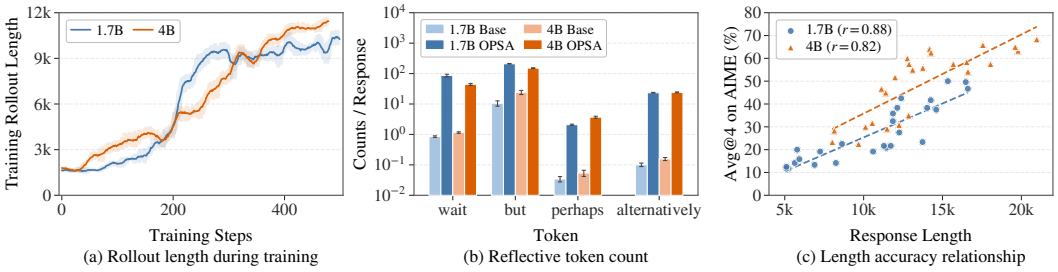


Figure 8: OPSA elicits long-form reasoning by generating more reflective tokens compared to base models. Moreover, AIME24 Avg@4 performance increases positively with response length.

Section 4.3 shows that, at high-entropy positions, OPSA redistributes probability mass among head tokens rather than concentrating it on a single prediction. Such positions often correspond to "fork" tokens in the chain of thought (Wang et al., 2025; Zhang et al., 2026), where different tokens may initiate distinct reasoning branches. Fig. 8(a) shows that the response length increases steadily throughout OPSA training. To better understand this change, we compare the frequency of reflective tokens in reasoning trajectories before and after training. As shown in Fig. 8(b), the OPSA-trained model produces substantially more reflective expressions, indicating more frequent reflection and self-correction during reasoning. We further observe a clear positive correlation between response length and answer accuracy. Together, these findings suggest that OPSA promotes exploration at high-entropy forks, leading the model toward longer and potentially more productive reasoning branches characterized by increased reflection and self-correction.

Performance Drops When Fork Tokens Are Masked. To validate whether OPSA's gains primarily arise from probability redistribution at fork tokens in the chain of thought, we conduct an ablation study by masking fork positions whose head-token sets contain reflective words, thereby excluding these positions from training. As shown in Fig. 9, masking these fork tokens largely eliminates the increases in both response length and accuracy observed under OPSA. Moreover, the response length collapses at approximately 300 training steps. These results demonstrate that applying OPSA at fork positions suppresses low-probability tail tokens while redistributing probability mass among competing head tokens. This encourages the policy to explore longer and more reflective reasoning branches, which appears to be a major source of its performance improvement.

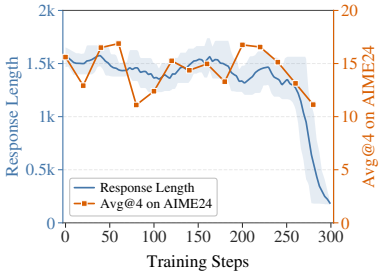


Figure 9: Training dynamics of OPSA when masking fork tokens.

Reflective Word Set for Experiments in Fig. 9. We list the detailed reflective word set used in experiments in Fig. 9 as follows:

Reflective Word Set

```
"wait", "however", "but", "alternatively", "hmm", "perhaps",
"check", "might", "actually"
```

Following the standard OPSA training procedure, we first identify the 20% of token positions with the lowest student log-probabilities. At each selected position, we inspect the student's top-5 candidate tokens. If any candidate belongs to the predefined reflective word set, we classify the position as a reflective fork and exclude it from the training objective.

C.2 Ablation Study of The Fraction of Trained Tokens.

We have examined the advantage-assignment scheme of OPSA in Section 3.2, where negative advantages positively correlated with token entropy yield the strongest improvement. Here, we ablate the proportion of low-log-probability tokens used for training. Fig. 10 reports the Avg@4 training curves when OPSA is applied to the lowest 10%, 20% (ours), 30%, and 40% of tokens ranked by their on-policy log-probabilities.

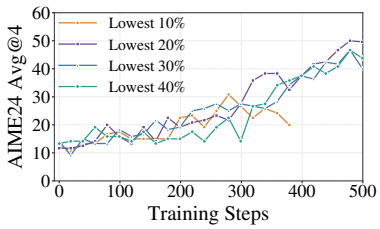


Figure 10: Ablation study of OPSA with different training token ratios.

Training on only the bottom 10% performs substantially worse than the other settings. We find that these tokens consist almost entirely of low-probability tail tokens outside the top-1 prediction. Applying negative advantages exclusively to them over-sharpens the policy distribution and causes a pronounced entropy decrease, thereby limiting further improvement. In contrast, the 20%, 30%, and 40% settings all increase Avg@4 to above 45. These results indicate that OPSA is not highly sensitive to the exact token-selection ratio and that training on only the bottom 20% of sampled tokens is sufficient to achieve substantial gains.

C.3 Performance of Training on Code Generation Tasks

In this section, we evaluate OPSA in a broader domain to further assess its applicability. Unlike the main experiments, which focus primarily on mathematical reasoning, we study OPSA training on code-generation tasks.

Models and Datasets. We train OPSA on two model families with different reasoning modes, using `Qwen3-4B` in non-thinking mode and `Olmo3-7B-Instruct`. Training hyperparameters follow the settings described in Appendix B.2. We use Open-R1 Codeforces (Penedo et al., 2025) as the training dataset and select the best checkpoint based on MBPP+ validation performance. Final in-domain evaluation is conducted on LiveCodeBench v6 and HumanEval+, while out-of-domain generalization is evaluated on the mathematical reasoning benchmark AIME24 and the general question-answering benchmark GPQA-Diamond (Rein et al., 2023).

Main Results. Table 8 shows that OPSA consistently improves both in-domain code-generation performance and out-of-domain generalization across the two model families. On `Qwen3-4B`, OPSA improves Avg@32 by 11.86 points on LCB v6 and 7.03 points on HumanEval+, with corresponding Pass@32 gains of 14.86 and 3.66 points. Similar improvements are observed on `Olmo3-7B-Instruct`, where Avg@32 increases by 7.95 points on LCB v6 and 2.10 points on HumanEval+. These results demonstrate that OPSA remains effective when trained directly on code-generation tasks. Interestingly, the gains also transfer beyond the training domain. On `Qwen3-4B`, OPSA improves AIME24 Avg@32 and Pass@32 by 5.63 and 10.00 points, respectively, and increases GPQA-Diamond Avg@32 by 2.58 points. `Olmo3-7B-Instruct` shows the same trend, with consistent improvements on both AIME24 and GPQA-Diamond. These results further demonstrate the robustness of OPSA across training domains, model families, and reasoning modes.

Table 8: Performance of OPSA across different models on in-domain code generation and out-of-domain mathematical and general Q&A tasks.

Methods	Code (I.D.)				O.O.D.		
	LCB v6		HumanEval+		AIME24		GPQA_D
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32
Qwen3-4B	25.71	37.14	78.75	93.90	23.33	56.67	38.46
w/ OPSA	37.57	52.00	85.79	97.56	28.96	66.67	41.04
Δ	+11.86	+14.86	+7.03	+3.66	+5.63	+10.00	+2.58
	↑ 46.1%	↑ 40.0%	↑ 8.9%	↑ 3.9%	↑ 24.1%	↑ 17.6%	↑ 6.7%
Olmo3-7B-Instruct	24.14	49.14	81.52	96.95	54.48	83.33	34.12
w/ OPSA	32.09	55.43	83.62	98.17	56.25	86.67	35.66
Δ	+7.95	+6.29	+2.10	+1.22	+1.77	+3.34	+1.54
	↑ 32.9%	↑ 12.8%	↑ 2.6%	↑ 1.3%	↑ 3.2%	↑ 4.0%	↑ 4.5%

C.4 Multi-Domain On-Policy Self-Adaptation

Table 9: Performance comparison of single-domain OPSA ($\text{OPSA}_{\text{math}}$ and $\text{OPSA}_{\text{code}}$) and multi-domain OPSA (MD-OPSA) across math and code benchmarks. **Bold** and underlined values indicate the best and second-best results, respectively. Training and evaluation use non-thinking mode.

Methods	Math						Code			
	AIME24		AIME25		HMMT25		LCB v6		HumanEval+	
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32
Qwen3-4B	23.33	56.67	20.52	56.67	13.13	33.33	25.71	37.14	78.75	93.90
w/ $\text{OPSA}_{\text{math}}$	62.08	**83.33**	58.44	**83.33**	**37.40**	60.00	27.95	45.71	83.35	96.34
w/ $\text{OPSA}_{\text{code}}$	28.96	66.67	22.92	63.33	13.85	36.67	**37.57**	52.00	**85.79**	97.56
w/ MD-OPSA	**66.77**	**83.33**	**60.83**	80.00	36.88	**70.00**	36.09	**56.00**	84.96	**98.78**

Multi-Teacher On-Policy Distillation (MOPD) (Ma et al., 2026) has been shown to enable effective capability merging during model post-training. In our previous experiments, we also found that OPSA trained on data from different domains achieves strong performance on both in-domain and out-of-domain evaluations. In particular, the substantial in-domain gains suggest that capability improvement may arise from reshaping the policy distribution through training on domain-specific data. This naturally raises a question: can we eliminate the domain-specific teacher models required by MOPD and instead achieve substantial multi-domain capability improvements by allowing the student policy to reshape its own distribution across data from different domains?

To investigate this question, we introduce **M**ulti-**D**omain **O**n-**P**olicy **S**elf-**A**daptation (**MD-OPSA**). MD-OPSA requires neither modifications to the training objective nor domain-specific routing or credit assignment. We simply randomly mix training data from different domains and optimize the policy using the same OPSA objective in Eq. 5. More specifically, we randomly sample an equal number of math and code-generation prompts from DAPO-17k and Open-R1 Codeforces, respectively, which are the same training datasets used in our previous experiments. We train `Qwen3-4B` on the mixed dataset, with non-thinking mode enabled for both training rollouts and evaluation.

Table 9 compares MD-OPSA with OPSA trained separately on math and code data. As expected, single-domain training primarily improves performance within its own domain: $\text{OPSA}_{\text{math}}$ yields substantial gains on mathematical reasoning, while $\text{OPSA}_{\text{code}}$ performs best on several code-generation metrics. In contrast, MD-OPSA successfully combines the benefits of both domains, achieving strong improvements across all math and code benchmarks without domain-specific teachers, routing, or credit assignment. Notably, MD-OPSA even surpasses the corresponding single-domain specialists on several metrics, including Avg@32 on AIME24 and AIME25 and Pass@32 on LCB v6 and HumanEval+. On the remaining benchmarks, its performance remains close to the best single-domain model. These results suggest that simply mixing multi-domain data allows OPSA to reshape the student policy toward capabilities from multiple domains with little apparent

interference, providing a simple teacher-free alternative for capability merging. Interestingly, the effects of distribution reshaping across domains may not be entirely orthogonal: adaptation on one domain can also provide transferable benefits to others, which may help explain the strong cross-domain performance of MD-OPSA.

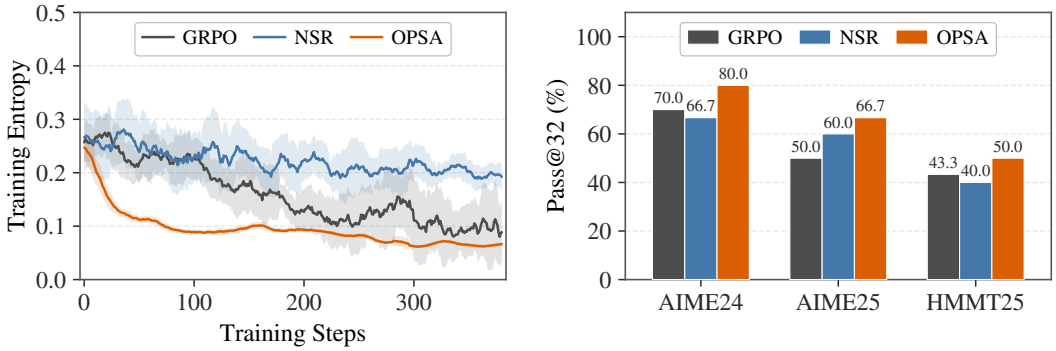
C.5 Comparison to Negative Sample Reinforcement

Table 10: Comparison of different on-policy RL methods on `Qwen3-1.7B`.

Methods	AIME24		AIME25		HMMT25		Average	
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32
Qwen3-1.7B	13.44	40.00	9.69	30.00	5.73	23.33	9.62	31.11
+ GRPO	33.96	70.00	25.31	50.00	15.10	43.33	24.79	54.44
+ NSR	32.08	73.33	24.17	60.00	16.15	43.33	24.13	58.89
+ OPSA	48.85	80.00	35.31	66.67	23.33	50.00	35.83	65.56

NSR (Zhu et al., 2025a) decomposes RLVR trajectories into correct and incorrect samples and performs trajectory-level reinforcement learning with negative advantages. Specifically, it assigns a fixed negative advantage to every token in an incorrect trajectory, thereby improving exploration while maintaining stable policy entropy during training. The key distinction between OPSA and NSR is that OPSA requires no supervisory signal, including verifiable rewards. Instead, it assigns token-level negative advantages to low-log-probability tokens across all trajectories and dynamically modulates these advantages according to token entropy, enabling substantially finer-grained optimization. Combined with our analysis in Section 4.3, these results suggest that OPSA suppresses low-probability tail tokens and redistributes their probability mass more evenly among high-probability head tokens. This mechanism preserves accuracy at low-entropy positions while enhancing exploration at high-entropy fork tokens. As shown in Table 10, OPSA consistently outperforms both GRPO and NSR in single-sample accuracy and pass@k, validating the effectiveness of our approach. The results further demonstrate that applying negative advantages to low-probability tokens even within correct trajectories is important for improving model performance.

C.6 Low Entropy Can Still Support Effective Exploration



(a) Training Entropy

(b) Pass@32 Performance

Figure 11: Training dynamics of different methods on `Qwen3-1.7B` and pass@32 performance.

As shown in Figure 11, OPSA exhibits substantially lower training entropy than GRPO and NSR while consistently achieving higher pass@32 on AIME24, AIME25, and HMMT25. This result demonstrates that aggregate entropy alone is not a reliable indicator of a model's exploration ability. Effective exploration depends more critically on how uncertainty is allocated across token positions. At high-entropy fork tokens, OPSA maintains a relatively balanced probability distribution among plausible head tokens, enabling the model to explore multiple promising reasoning branches.

At low-entropy positions, it further concentrates probability mass on high-confidence tokens, improving sampling accuracy and preventing trajectories from entering implausible, low-confidence branches. Consequently, OPSA achieves lower overall entropy without sacrificing diversity and meaningful exploration at critical decision points, consistent with our analysis in Section 5.2.

C.7 Performance Comparison Under Similar Token Budgets

Table 11: AIME24 performance of different RL strategy on similar inference token budgets.

Methods	Token Counts	Avg@32
Qwen3-1.7B	4457	13.44
+ GRPO	19108	33.96
+ GRPO ("wait")	23261	32.81
+ OPD	15286	32.08
+ OPD ("wait")	23472	31.67
+ OPSA	23205	48.85

To verify that OPSA's gains are not merely a consequence of generating longer responses, we construct a token-budget-matched inference control for the GRPO and OPD baselines. Specifically, we first generate a baseline response, remove its terminal `\boxed{}` answer, append a single "wait" token to the remaining reasoning trace, and then resume decoding. A minimum generation-length constraint is applied so that the resulting responses approximately match the average response length of the OPSA-trained model. We evaluate the first complete `\boxed{}` answer produced after reaching the target token budget while keeping all other decoding settings unchanged. As shown in Table 11, allocating additional inference-time tokens does not improve the baselines, and OPSA continues to substantially outperform both GRPO and OPD under comparable response lengths. These results indicate that OPSA's improvements cannot be explained by response length alone. Instead, they are consistent with OPSA reshaping the policy distribution to improve precision at low-entropy positions and encourage exploration at high-entropy positions, thereby steering generation toward more effective reflective reasoning branches.

C.8 OPSA as a Cold-Starting for GRPO Training

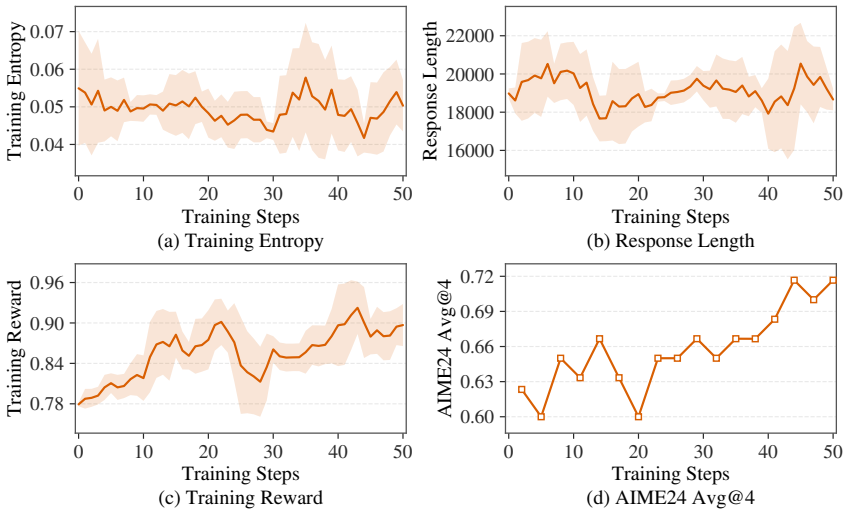
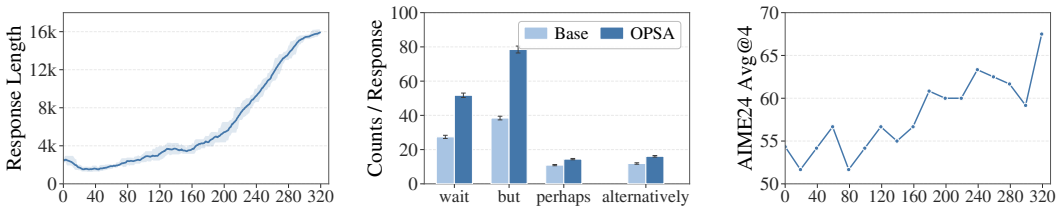


Figure 12: Training dynamics of GRPO training started from the OPSA cold-starting 4B model.

OPSA demonstrates strong effectiveness during post-training, improving both accuracy and Pass@k while preserving sampling diversity. In this section, we investigate whether OPSA, as an external-supervision-free method, can serve as a "free-lunch" cold start before further RL training and enable

additional performance gains. We conduct experiments on `Qwen3-4B`, using an OPSA checkpoint as the initialization and further training it with GRPO on DAPO-17k. Figure 12 reports the resulting training dynamics. On the validation set, Avg@4 continues to improve steadily, increasing by approximately 9 points after 40 training steps, while the training curve remains smooth without signs of collapse. These results suggest that OPSA can provide an effective cold start for subsequent RL training, further extending the model's capabilities.

C.9 OPSA Elicits Reasoning Capabilities from an Instruct Model



(a) Training response length (b) Reflective token count (c) Validation accuracy

Figure 13: Training dynamics of OPSA on Olmo3-7B-Instruct

Our previous analysis in Section C.1 suggests that OPSA improves reasoning by steering the policy toward longer and more reflective reasoning branches. Since Qwen3 is a hybrid model with an explicit thinking mode, it remains unclear whether this behavior simply reflects the recovery of reasoning patterns already encoded in the model. To examine whether OPSA can induce similar behavior beyond hybrid models, we further apply OPSA to `Olmo3-7B-Instruct`. As shown in Fig. 13, OPSA steadily increases response length together with validation accuracy, while substantially increasing the frequency of reflective tokens such as "wait" and "but". These results suggest that OPSA does not merely recover a pre-existing thinking mode. Instead, by reshaping the policy distribution at high-entropy fork positions, it increases the likelihood of exploring reflective reasoning branches and can elicit stronger reasoning behavior even from an instruct model.

C.10 Tuning Inference-Time Sampling Hyperparameters Can't Replace OPSA

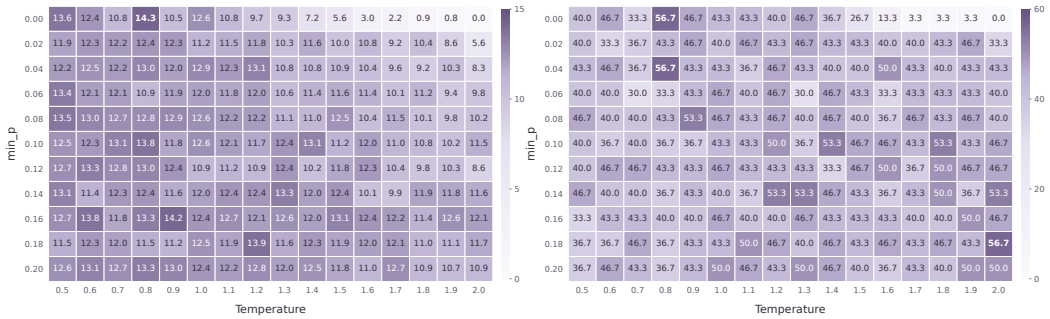


Figure 14: Results on AIME24 with `Qwen3-1.7B` in non-thinking mode under different min-p and temperature settings: (a) Avg@32 and (b) Pass@32. All values are percentages, and **bold** entries indicate the best result in each panel.

Although tuning inference-time sampling parameters can also suppress low-probability tail tokens and redistribute sampling probability among head tokens, Fig. 14 shows that it cannot reproduce the gains of OPSA. We attribute this gap to two key differences. First, inference-time sampling applies a fixed transformation to the base policy at every decoding step without changing the underlying policy itself. However, the desired amount of tail suppression and head-token redistribution can vary substantially across token positions and reasoning states, making it difficult to determine a

single optimal temperature or min-p value in advance. In contrast, OPSA adapts the magnitude of its updates according to the policy's token-level uncertainty, enabling state-dependent distribution reshaping through training rather than relying on a globally fixed sampling rule. Second, OPSA performs this reshaping iteratively: each policy update changes the distribution from which subsequent trajectories are sampled, exposing new reasoning branches that can be further optimized. Such parameter updates can also generalize across contexts and alter the modes of the token distribution, while inference-time sampling continues to operate on the same underlying base-policy modes. Thus, although both approaches can suppress tail tokens, OPSA learns a new reasoning policy rather than merely sampling differently from the original one.