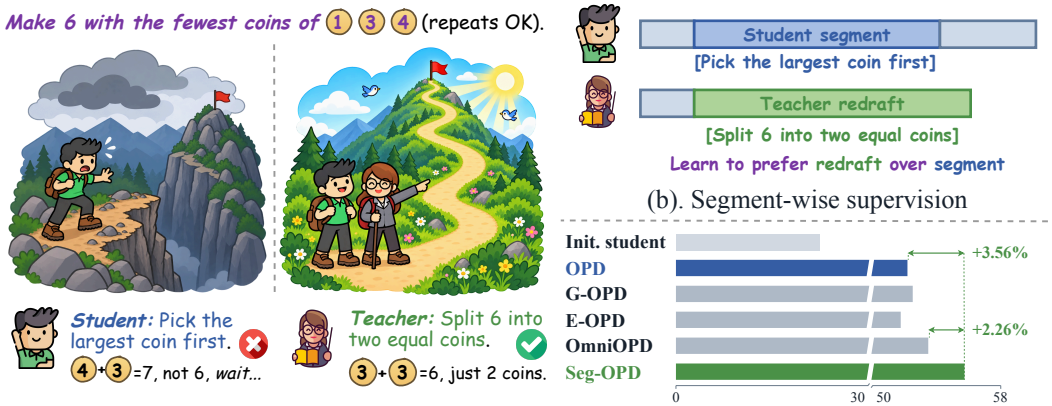
LEARNING TO REVISE REASONING WITH SEGMENT-WISE ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) improves large language model reasoning by training students on their own rollouts with dense token-wise supervision from the teacher. However, token-wise OPD does not explicitly provide a coherent alternative reasoning step showing how the student's step could be revised to improve subsequent reasoning. Furthermore, this paradigm can become less effective when the student produces a degenerate reasoning prefix, as subsequent teacher supervision remains conditioned on that prefix and may reinforce poor reasoning patterns. In this work, we focus on learning reasoning revision with segment-wise OPD to rework intermediate reasoning steps and better support subsequent reasoning. Through controlled reasoning interventions, we find that replacing student segments with teacher redrafts improves subsequent reasoning accuracy. Therefore, we address the problem of turning teacher redrafts into explicit supervision for learning to revise reasoning. We propose Segment-wise On-Policy Distillation (SEG-OPD), which selects student segments based on an uncertainty metric and obtains corresponding teacher redrafts. SEG-OPD trains the student to prefer teacher redrafts over their paired student segments while retaining dense token-wise OPD supervision. Extensive experiments on mathematical reasoning and competitive programming tasks show that SEG-OPD-trained students achieve higher revision success rates than baselines. SEG-OPD consistently outperforms the compared state-of-the-art baselines in reasoning accuracy with an average relative improvement of 5.22% across diverse models and tasks. Code is available at https://anonymous.4open.science/r/Seg-OPD.



(a). An example of reasoning revision

(b). Segment-wise supervision

(c). Overall performance

Figure 1: **(a). An example of reasoning revision.** The student's largest-coin-first attempt overshoots the target, whereas the teacher redraft reaches it with two equal-value coins. **(b). Segment-wise supervision.** The paired student segment and teacher redraft illustrate explicit supervision for learning reasoning revision. **(c). Overall Performance.** Reasoning accuracy (%) of SEG-OPD and baselines for a Qwen3-14B student on AIME25.

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

On-Policy Distillation (OPD) has emerged as an effective paradigm for LLM post-training (Agarwal et al., 2024). OPD has the student generate its own reasoning trajectories and prompts a stronger teacher to provide feedback. By integrating the dense token-wise supervision of Supervised Fine-Tuning (SFT) into the on-policy exploration of Reinforcement Learning (RL), OPD mitigates both the distribution shift introduced by imitating offline demonstrations in SFT (Ouyang et al., 2022) and the sparse reward signals of outcome-based RL (Shao et al., 2024).

Despite these strengths, OPD suffers from two inherent limitations. *First,* because vanilla OPD provides only local corrections through token-wise supervision, the student is never explicitly shown the coherent reasoning step that spans multiple tokens. In such cases, local corrections do not directly show the student how to revise the reasoning step as a whole and redirect subsequent reasoning. *Second,* due to LLMs' autoregressive nature, their subsequent reasoning is conditioned on the prefix. Therefore, the existing degenerate prefix from the student rollout can be harmful and thus degrade the teacher's guidance, leading to suboptimal student learning. Recent studies show that on degenerate prefixes, teacher feedback can favor unproductive continuations rather than penalize them, potentially reinforcing patterns such as repetition loops (Luo et al., 2026; Fu et al., 2026).

These limitations motivate us to improve vanilla OPD by replacing unproductive reasoning steps and redirecting the student's subsequent reasoning. More precisely, given a prefix and a student reasoning segment, the teacher should provide a coherent redraft that revises the corresponding segment while preserving the prefix. Such a redraft retains the on-policy context that exposes the student's current unproductive reasoning mode, providing an alternative reasoning transition that can redirect subsequent reasoning. We use **reasoning revision** to describe reworking an intermediate reasoning step to better support subsequent reasoning, and illustrate it in Fig. 1. In Fig. 1(a), the teacher redraft offers an equal-split strategy instead of the student's largest-coin-first approach. Fig. 1(b) illustrates the use of teacher redrafts as explicit supervision for learning reasoning revision.

Though promising, implementing reasoning revision learning faces two challenges. *First,* teacher redrafting is costly, so how can we fully utilize the limited budget to select student segments worth revising and obtain suitable teacher redrafts? *Second,* how can the learning objective exploit the correspondence between student segments and teacher redrafts while preserving the dense token-wise supervision that makes OPD effective? To address these challenges, we conduct a preliminary test-time analysis in Sec. 3 to inform our method's design. We reveal that replacing selected student segments with teacher redrafts improves subsequent reasoning accuracy for both the initial and OPD-trained student models. Furthermore, we find that student reasoning pivots associated with high uncertainty provide more useful revision targets than random anchors.

Building upon these insights, we propose a novel Segment-wise On-Policy Distillation method (SEG-OPD). To better utilize the limited budget, SEG-OPD uses an entropy-based uncertainty metric to guide student segment selection and obtains corresponding teacher redrafts, providing targeted revision supervision. SEG-OPD employs a revision learning objective that encourages the student to prefer teacher redrafts over the paired student segments through a segment-wise preference loss, while retaining dense token-wise OPD supervision. As Fig. 1(c) shows, SEG-OPD outperforms the strongest baseline by 2.26% in reasoning accuracy with Qwen3-14B on AIME25.

We summarize our contributions as follows:

- We formulate the problem of reasoning revision in OPD as learning from teacher redrafts of students' own intermediate reasoning. Through controlled trajectory interventions, we show that replacing student segments with teacher redrafts substantially improves downstream reasoning, establishing teacher redrafts as useful explicit revision supervision;
- We introduce SEG-OPD, which enables students to learn reasoning revision from revision units that pair their own reasoning segments with corresponding teacher redrafts. To achieve this, SEG-OPD trains students to prefer the redrafts over their paired student segments while retaining dense token-wise OPD feedback;
- We conduct experiments with diverse student models across a wide range of math and code benchmarks. SEG-OPD outperforms the compared state-of-the-art OPD methods in reasoning accuracy, achieving a 5.22% average relative improvement over competitors. Beyond accuracy, SEG-OPD also achieves higher reasoning revision success rates and fewer reasoning loops after training.

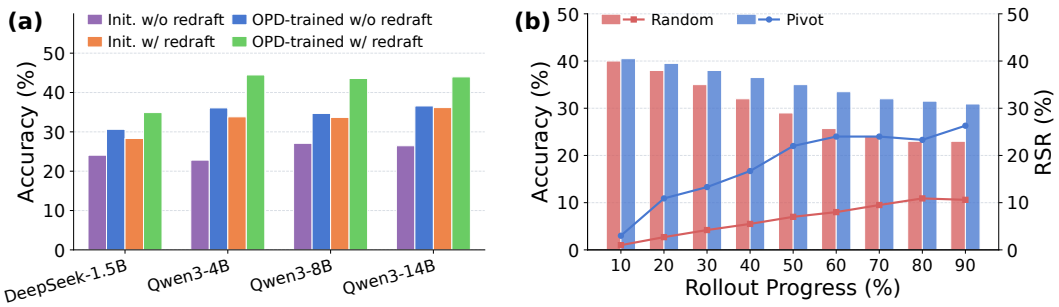


Figure 2: **Effectiveness of teacher redrafts and segment selection. (a).** Reasoning accuracy of initial/OPD-trained students with/without teacher redrafts on AIME24. **(b).** After replacing segments selected with random/pivot anchors with teacher redrafts, the reasoning accuracy (bars, left axis) and revision success rate (lines, right axis) of Qwen3-4B on AIME26.

2 Related Work

2.1 On-Policy Distillation

On-policy Distillation (OPD) trains students on their own rollouts using teacher supervision (Agarwal et al., 2024). Recently, OPD has been extended to multi-teacher, multi-task post-training (Xiao et al., 2026; Xu et al., 2026a) and on-policy self-distillation (Zhao et al., 2026a;b). Although OPD's effectiveness primarily depends on learning the teacher's reasoning behavior (Li et al., 2026; Luo et al., 2026), how these patterns transfer remains underexplored (Fu et al., 2026; Yang et al., 2026; Cai et al., 2026). By explicitly guiding the student to prefer the teacher's redraft over its own segment, SEG-OPD emphasizes this reasoning behavior during training.

2.2 Learning from Reasoning Interventions

As LLMs reason autoregressively, early errors can misdirect subsequent steps, motivating interventions (Lai et al., 2024; Xu et al., 2026b). Prior work trains reward models with step-level labels (Lightman et al., 2024; Wang et al., 2024), compares reasoning steps for preference learning (Lai et al., 2024; Zhang et al., 2024; Yang et al., 2025b), or introduces teacher guidance during on-policy training (Xu et al., 2026b; Huang et al., 2026; Zhao et al., 2026b). Unlike offline step-pairing (Lai et al., 2024; Zhang et al., 2024) or interleaving teacher generations within a rollout (Xu et al., 2026b), SEG-OPD pairs student segments with teacher redrafts, training the student to prefer these redrafts over its own segments while retaining token-wise OPD on the remaining tokens.

3 Preliminary Analysis

In this section, we present the preliminaries on reasoning revision in on-policy distillation and empirical cases that motivate the methodology of our SEG-OPD.

3.1 Notations

Let $x = (x_1, x_2, \ldots, x_n)$ be an input prompt and $y = (y_1, y_2, \ldots, y_m)$ a response. We denote the prefix up to step t by $q_t = (x, y_{<t})$, where $y_{<t} \triangleq (y_1, y_2, \ldots, y_{t-1})$. OPD setting generally involves two LLMs: a trainable student π_θ and a fixed teacher π_*.

3.2 On-Policy Distillation

By exploring its own action space and receiving dense feedback from the teacher distribution π_*, OPD enables the student π_θ to learn from its own rollouts as:

$$\mathcal{J}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_*(\cdot \mid x))\right]. \tag{1}$$

For $x \sim \mathcal{D}$ and $y \sim \pi_\theta(\cdot \mid x)$, the gradient of Eq. (1) can be written as:

$$\nabla_\theta \mathcal{J}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x,y}\left[\sum_{t=1}^{m}\left(\sum_{t'=t}^{m}\left[\log \pi_\theta(y_{t'} \mid q_{t'}) - \log \pi_*(y_{t'} \mid q_{t'})\right]\right)\nabla_\theta \log \pi_\theta(y_t \mid q_t)\right]. \quad (2)$$

In practice, OPD methods use a discount factor of zero to focus on next-token optimization and thus approximate the gradient in Eq. 2 as:

$$\nabla_\theta \mathcal{J}_{\mathrm{OPD}}(\theta) \approx \mathbb{E}_{x,y}\left[\sum_{t=1}^{m}\left[\log \pi_\theta(y_t \mid q_t) - \log \pi_*(y_t \mid q_t)\right]\nabla_\theta \log \pi_\theta(y_t \mid q_t)\right]. \quad (3)$$

The approximation in Eq. 3 is commonly called token-wise OPD. Although computationally efficient, token-wise OPD provides feedback only for individual tokens, without explicitly presenting a coherent alternative reasoning step as a revision target. We therefore consider teacher redrafts as explicit revision targets for student segments. The segment-wise preference objective of SEG-OPD uses conditional joint probabilities to compare each student segment with its teacher redraft. To assess the value of these revision targets, we first examine whether replacing a student segment with a teacher redraft at test time improves subsequent reasoning accuracy.

3.3 UNDERSTANDING REASONING REVISION IN OPD

Teacher Redrafts Can Redirect Subsequent Reasoning. As LLMs generate responses autoregressively, each reasoning token is conditioned on the preceding context. This dependency suggests that replacing an incorrect intermediate step with a better redraft can redirect subsequent reasoning. To validate this, we first manually locate the wrong reasoning segment s in the student rollout, and then assign the same prefix q to a teacher model to generate a redraft r. For comparison, we prompt the student to continue the subsequent reasoning from (q, s) and (q, r), respectively. As shown in Fig. 2(a), replacing s with r increases reasoning accuracy for both initial and OPD-trained student models, indicating that teacher redrafts can provide direct and effective reasoning revisions.

Reasoning Pivots Provide Better Revision Targets. To extend these benefits to on-policy training, we need an efficient way to select informative segments worth revising without relying on manual review. Prior work shows that high-entropy tokens indicate potential turning points in reasoning (Wang et al., 2025b; Cheng et al., 2026). Therefore, we select these pivots as anchors and expand them in both directions to period-marked sentence boundaries, forming a student segment for teacher redrafting. To examine whether reasoning pivots provide better revision targets, we compare reasoning units centered on **(i) random anchors** and **(ii) pivot anchors** around the same rollout positions. Beyond reasoning accuracy, we also report the revision success rate (RSR), defined as the proportion of student revision attempts in the complete trajectory that correct an earlier error with a substantively different alternative. For RSR evaluation, we call GPT-5.6 to extract reasoning steps from rollouts and assess whether the attempts meet this criterion. Appendix A.1 provides more details. As shown in Fig. 2(b), pivot anchors yield higher reasoning accuracy and higher student revision success rates than random anchors at matched rollout positions.

Summary. These results show that **(i)** replacing selected student segments with teacher redrafts can improve the student's subsequent reasoning accuracy; and **(ii)** at matched rollout positions, pivot anchors yield higher reasoning accuracy than random anchors and are accompanied by higher RSR. The analysis further motivates a novel segment-wise OPD paradigm in which the student directly learns reasoning revisions from teacher redrafts on selected segments.

4 METHODOLOGY

Our preliminary analysis shows that teacher redraft can redirect students' subsequent reasoning, and reasoning pivots provide better revision targets. These findings leave two open questions for learning reasoning revision: **(i)** How to select student segments and construct corresponding teacher redrafts for learning reasoning revision? **(ii)** How to derive the supervision signal from the revision unit to guide the student in learning reasoning revision?

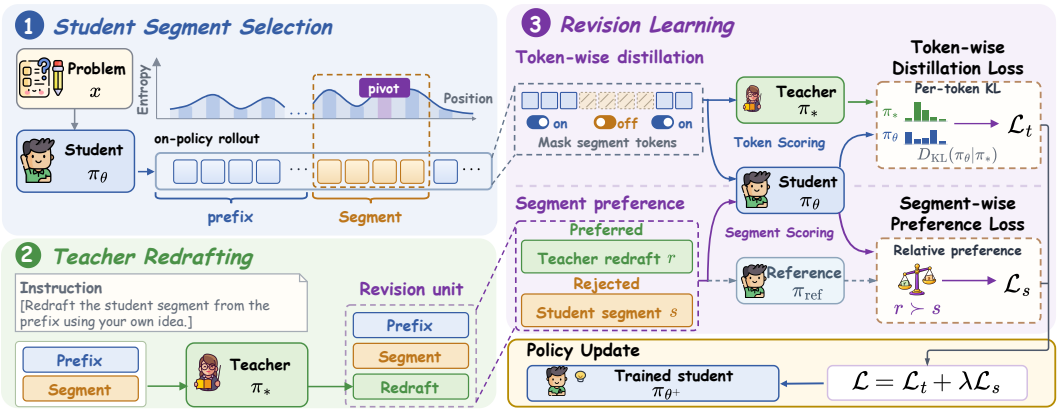


Figure 3: **Framework of SEG-OPD.** SEG-OPD enables students to learn reasoning revision by pairing student segments with teacher redrafts and using preference learning to favor these alternative reasoning steps over the corresponding student segments. Simultaneously, SEG-OPD retains dense token-wise supervision outside the selected segments.

To address these challenges, we propose SEG-OPD, and illustrate its framework in Fig. 3. As Fig. 3 shows, SEG-OPD first selects student segments around reasoning pivots in student rollouts. For each selected segment, SEG-OPD prompts the teacher to propose an alternative reasoning step under the same prefix, forming a revision unit from the prefix, student segment, and teacher redraft. Consequently, SEG-OPD trains the student by optimizing a segment-wise preference learning objective on these revision units while retaining token-wise OPD on the remaining valid rollout tokens. Next, we present each component in detail.

4.1 REVISION UNIT CONSTRUCTION

Segment Selection. Motivated by the analysis in Sec. 3.3, we find that student segments constructed using reasoning pivots achieve higher revision success rates and higher reasoning accuracy after redrafting than those constructed using random anchors at the same rollout positions. Since rollouts ending in incorrect results are more likely to contain reasoning steps that can benefit from revision, SEG-OPD restricts candidate segment construction to these rollouts.

For each selected rollout y, we design a hierarchical uncertainty metric based on two considerations: **(i)** a pivot anchor should exhibit high predictive uncertainty under the current student; and **(ii)** the complete reasoning segment containing the anchor should exhibit high aggregated uncertainty, rather than being selected because of an isolated high-entropy token. We first quantify uncertainty at each position using the predictive entropy recorded during rollout generation:

$$H_t = -\sum_{v \in \mathcal{V}} \pi_\theta(v \mid q_t) \log \pi_\theta(v \mid q_t), \tag{4}$$

where $\mathcal{V}$ denotes the token vocabulary. Then SEG-OPD selects the positions with the largest entropy values as pivot anchors and denotes the resulting set as T. For each pivot anchor $t_i \in T$, SEG-OPD expands it in both directions to the surrounding sentence boundaries marked by periods, yielding a candidate segment $s_i = y_{s_i^b:s_i^e}$, where s_i^b and s_i^e index its first and last tokens. Additional analysis on boundary construction is provided in Appendix B.1. We score each candidate by its mean predictive entropy penalized by the entropy dispersion within the segment:

$$U(s_i) = \frac{1}{|s_i|} \sum_{t=s_i^b}^{s_i^e} H_t - \delta_i, \quad |\mathcal{C}_K(y)| \le K, \tag{5}$$

where δ_i denotes the standard deviation of the token-entropy values within s_i, and $|s_i| = s_i^e - s_i^b + 1$ is its token count. Then, we greedily select segments in descending score order to form the set $\mathcal{C}_K(y)$, with at most K segments per rollout.

Teacher Redrafting. To provide a revision as a local alternative to the selected student segment for preference learning, SEG-OPD constructs a teacher redraft by revising the corresponding student reasoning step. We obtain this alternative from one full teacher reasoning, allowing the teacher to develop its reasoning toward the solution. To achieve this, SEG-OPD first provides the teacher model with both the prefix q and the student segment s. The teacher is explicitly prompted to rewrite s starting after q and continue reasoning; SEG-OPD then extracts the leading sentence as the teacher redraft r using the same boundary rule as for student segments, provided that the full draft passes a task-specific correctness check. The corresponding teacher redraft r, together with q and s, forms a revision unit $p = (q, s, r)$. Appendix A.2 provides more details on revision unit construction.

4.2 REASONING REVISION LEARNING

For each revision unit $p_i = (q_i, s_i, r_i)$, SEG-OPD trains the student to prefer r_i over s_i by optimizing a segment-wise preference objective while retaining token-wise OPD outside s_i.

Segment-Wise Preference Objective. We start with a preference learning objective similar to that used in prior work (Rafailov et al., 2023). Specifically, we use $r \succ s$ as the training preference relation between each teacher redraft and its paired student segment under the same prefix. Let π_{ref} be the base reference policy of the student. Following (Lu et al., 2024), we normalize each segment's log policy ratio by its token length and rescale the difference by the mean segment length, defining the revision margin as:

$$m_{\pi_\theta}(p) = \frac{|s| + |r|}{2} \left[\frac{1}{|r|} \log \frac{\pi_\theta(r \mid q)}{\pi_{\text{ref}}(r \mid q)} - \frac{1}{|s|} \log \frac{\pi_\theta(s \mid q)}{\pi_{\text{ref}}(s \mid q)} \right]. \tag{6}$$

Here, $\log \pi_\theta(r \mid q)$ and $\log \pi_\theta(s \mid q)$ are the logarithms of the segments' conditional joint probabilities, obtained by summing the token log probabilities over r and s, respectively. A positive revision margin in Eq. (6) means that the student's length-normalized log-probability gap between r and s exceeds that of the reference policy. Using the Bradley–Terry form (Bradley & Terry, 1952), we model the preference $r \succ s$ with probability $\sigma(\beta m_{\pi_\theta}(p))$. Let $\{p_i = (q_i, s_i, r_i)\}_{i=1}^N$ denote the N revision units in the current training batch. We take the negative log-likelihood of this probability and average over these units, which gives the segment-wise loss:

$$\mathcal{L}_s(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log \sigma(\beta m_{\pi_\theta}(p_i)). \tag{7}$$

Here, σ is the logistic sigmoid and $\beta > 0$ controls the margin scale. The loss assigns larger penalties to units with small or negative revision margins and decreases as the margin grows.

Token-wise Distillation. To help stabilize training (Grattafiori et al., 2024), SEG-OPD retains token-wise OPD alongside the segment-wise preference objective. The token-wise loss provides dense teacher supervision on the remaining valid rollout tokens. Specifically, for a sampled rollout $y = (y_1, \ldots, y_m)$, let $M_t \in \{0, 1\}$ denote the token mask. We set $M_t = 0$ when y_t lies in any student segment used in a revision unit and $M_t = 1$ otherwise. The mask only determines which positions are used in the token-wise loss and does not change the sampled rollout. Following the token-wise OPD formulation in Eq. (3), the token-wise loss with segment masking is:

$$\mathcal{L}_t(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^{m} M_t D_{\text{KL}}(\pi_\theta(\cdot \mid q_t) \,\|\, \pi_*(\cdot \mid q_t)) \right]. \tag{8}$$

Here, we present the final optimization function of SEG-OPD, based on the analysis above. We jointly optimize the token-wise distillation loss $\mathcal{L}_t$ and the segment-wise preference loss $\mathcal{L}_s$ as:

$$\theta^* = \arg\min_\theta \mathcal{L}_t(\theta) + \lambda \mathcal{L}_s(\theta), \tag{9}$$

where $\lambda > 0$ is the hyperparameter to control the contribution of the segment-wise preference loss. Appendices C and D detail SEG-OPD's training algorithm and complexity analysis, respectively.

Table 1: Main results on mathematical reasoning and competitive programming. We report the mean Pass@1 (%) across diverse student models and datasets. **Bold** and underline denote the best and second-best results per evaluation setting.

Method	Codeforces	TACO	HumanEval+	AIME24	AIME25	AIME26	MATH500	MINERVA
Student: DeepSeek-1.5B								
OPD	33.30	28.30	73.25	43.09	36.67	33.33	86.45	37.22
E-OPD	34.90	29.30	78.84	43.31	35.83	30.83	86.05	37.59
G-OPD	33.75	30.06	78.67	50.00	39.17	31.67	86.85	38.69
OmniOPD	33.90	29.70	79.39	42.50	32.50	28.33	86.05	37.04
SEG-OPD	**36.30**	**31.30**	**81.20**	**51.33**	**40.19**	**35.25**	**89.37**	**39.36**
Student: Qwen3-4B								
OPD	42.34	43.32	82.30	63.15	61.07	49.69	**90.71**	45.89
E-OPD	43.90	40.97	84.67	62.50	61.61	51.26	90.01	46.38
G-OPD	43.50	40.58	83.69	62.32	61.23	52.17	90.05	47.57
OmniOPD	40.04	38.68	82.62	63.50	59.56	50.49	88.73	44.85
SEG-OPD	**45.89**	**43.65**	**85.89**	**64.50**	**63.00**	**52.53**	89.10	**48.85**
Student: Qwen3-8B								
OPD	46.41	45.07	76.59	59.17	55.60	44.32	87.85	43.01
E-OPD	45.75	44.35	78.54	**63.68**	57.99	42.64	90.31	44.85
G-OPD	46.01	41.89	80.27	60.83	52.50	46.79	**90.40**	44.60
OmniOPD	46.86	**45.46**	78.51	55.38	57.08	46.60	88.30	43.66
SEG-OPD	**48.86**	45.35	**82.37**	63.37	**59.67**	**47.19**	88.36	**45.91**
Student: Qwen3-14B								
OPD	37.35	45.14	81.06	60.67	52.17	46.67	87.50	45.60
E-OPD	36.17	45.81	81.59	61.17	51.75	**49.34**	87.84	44.12
G-OPD	36.59	45.71	82.07	60.83	52.50	48.97	87.40	**47.22**
OmniOPD	39.88	**48.12**	82.69	59.97	53.47	45.76	**88.97**	45.23
SEG-OPD	**41.59**	47.68	**83.37**	**63.33**	**55.73**	49.17	88.40	46.00

5 EXPERIMENT

In this section, we conduct extensive experiments to answer the following research questions:

- **RQ1:** How does SEG-OPD perform against state-of-the-art baselines for on-policy distillation?
- **RQ2:** How does SEG-OPD affect the student's reasoning behavior?
- **RQ3:** How effective are the proposed components of SEG-OPD?

5.1 EXPERIMENT SETUP

Models We use DeepSeek-1.5B (Guo et al., 2025), Qwen3-4B, Qwen3-8B, and Qwen3-14B (Yang et al., 2025a) as students. Each student is paired with an RL-trained teacher from the same model family and parameter scale. Appendix A.3 provides more details.

Datasets We evaluate SEG-OPD on mathematical reasoning and competitive programming. For mathematical reasoning, we train on DeepMath-103K (He et al., 2025b) and evaluate on AIME24, AIME25, AIME26 (Cai et al., 2026), MATH500 (Lightman et al., 2024), and MINERVA (Lewkowycz et al., 2022). For competitive programming, we train on Eurus-RL-Code, evaluate on Codeforces, TACO (Cui et al., 2025), and HumanEval+ (Liu et al., 2023). Appendix A.4 provides more details.

Baselines We compare SEG-OPD against OPD (Agarwal et al., 2024), G-OPD with reward extrapolation (Yang et al., 2026), E-OPD, which adds a forward-KL constraint at the teacher's uncertain positions (Jin et al., 2026), and OmniOPD (Zhou et al., 2026), which supervises student chunks through similarity with teacher continuations. Appendix A.5 provides more details.

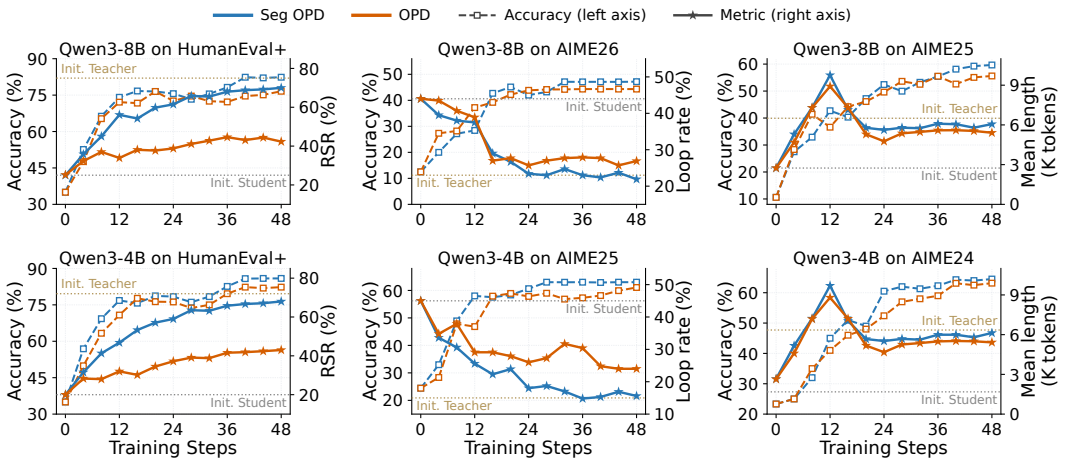


Figure 4: **Reasoning accuracy** (dashed lines, left axis) and **behavioral metric statistics** (solid lines, right axis). Gray and gold horizontal lines indicate the initial behavioral metric values for the student and teacher models, respectively.

5.2 SEGMENT-WISE OPD PERFORMANCE

To answer **RQ1**, we compare SEG-OPD with competitive baselines on mathematical reasoning and competitive programming. Results in Tab 1 demonstrate that SEG-OPD achieves an average relative accuracy gain of 5.22% over baselines across all models and datasets. Specifically, we observe that:

- Compared with standard OPD, SEG-OPD improves reasoning accuracy in most settings, with gains of up to 8.24% for DeepSeek-1.5B on AIME24. This improvement demonstrates the effectiveness of SEG-OPD's segment-wise preference learning.
- SEG-OPD outperforms E-OPD and G-OPD in most settings. Using student uncertainty, SEG-OPD selects segments around reasoning pivots. It obtains teacher redrafts that offer alternative reasoning paths, giving students examples of how to redirect reasoning at potential turning points, rather than requiring students to infer such changes from token-wise distillation signals alone.
- SEG-OPD also outperforms the closely related baseline OmniOPD in most settings, with gains of up to 8.83% for DeepSeek-1.5B on AIME24. OmniOPD weights student chunks by their similarity to teacher continuations, whereas SEG-OPD directly learns to prefer teacher redrafts over their paired student segments.

5.3 ANALYSIS OF STUDENT REASONING BEHAVIOR

To answer **RQ2**, we compare SEG-OPD with OPD, using Qwen3-8B and Qwen3-4B as students, and examine their reasoning behavior during training. Alongside reasoning accuracy, we report the student's revision success rate (**RSR**) with the same definition as in Sec. 3.3, the **loop rate**, defined as the proportion of steps that repeat earlier reasoning without progress, and **mean length**, the average number of tokens per rollout. Appendix B.2 provides further analysis on student reasoning behavior. We make the following observations from Fig. 4:

- Higher RSR with SEG-OPD suggests that learning from teacher redrafts helps the student revise its own reasoning more successfully. For Qwen3-8B on HumanEval+, RSR peaks at 70.0%, compared with 44.6% for OPD and 75.0% for the initial teacher. Peak reasoning accuracy reaches 82.4% with SEG-OPD versus 76.6% with OPD.
- Lower loop rates accompany higher accuracy in training. For Qwen3-4B on AIME25, SEG-OPD achieves a 15.6% loop rate and 63.0% accuracy, versus 24.0% and 61.1% with OPD, suggesting that revision learning helps students move past stalled reasoning toward a solution.
- For Qwen3-8B on AIME25, SEG-OPD reaches 59.7% accuracy, compared with 55.6% for OPD, with mean response lengths of 6.1K and 5.4K tokens, respectively. Across both length comparisons, SEG-OPD achieves higher final accuracy with comparable mean response lengths.

Table 2: **Ablation studies** of SEG-OPD components with Qwen3-4B as the student.

Variant	Codeforces	TACO	HumanEval+	AIME24	AIME25	AIME26	MATH500	MINERVA
SEG-OPD	**45.89**	**43.65**	**85.89**	**64.50**	**63.00**	**52.53**	**89.10**	**48.85**
SEG-OPD \S	42.60	38.07	72.53	54.16	59.18	51.54	75.49	45.23
SEG-OPD \T	40.27	33.25	71.89	56.90	57.25	52.39	70.05	46.27
SEG-OPD \N	37.58	31.75	70.69	42.96	56.89	49.20	73.29	40.67

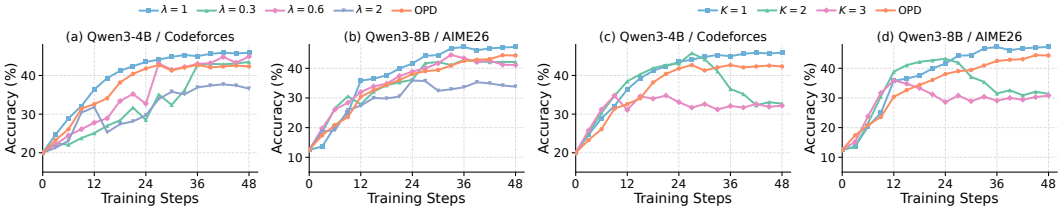


Figure 5: **Hyperparameter sensitivity analysis of SEG-OPD.**

5.4 Ablation Studies and Hyperparameter Sensitivity Analysis

Ablation Studies. To answer **RQ3**, we evaluate three variants of SEG-OPD: **(i)** We replace pivot anchors with random anchors to obtain SEG-OPD \S. **(ii)** We prompt the teacher in SEG-OPD \T to generate from the prefix q without providing s or explicit revision instruction. **(iii)** We replace the segment-wise preference loss $\mathcal{L}_s$ in Eq. (7) with supervised fine-tuning (SFT) loss on r to obtain SEG-OPD \N. From the results in Tab. 2, we observe that:

- Reasoning pivots provide more useful redrafting targets than position-matched random anchors, as shown by SEG-OPD gains over SEG-OPD \S.
- The comparison with SEG-OPD \T shows that teacher redrafts tailored to the student's reasoning outperform teacher continuations from the prefix alone.
- The advantage over SEG-OPD \N supports learning to prefer teacher redrafts over their paired student segments, rather than simply imitating the redrafts. Appendix B.3 provides further analysis on the training dynamics and performance of SEG-OPD and this variant.

Hyperparameter Analysis. We vary the segment-wise loss weight λ in Eq. (9) and the number K of student segments selected per rollout in Eq. 5. Fig. 5 shows sweeps over $\lambda \in \{0.3, 0.6, 1, 2\}$ and $K \in \{1, 2, 3\}$, with the default λ used in Tab. 1 normalized to 1. From the figure, we observe that:

- Smaller weights preserve accuracy, whereas $\lambda = 2$ degrades both settings, ending at 33.8% for Qwen3-8B on AIME26 versus 47.2% at $\lambda = 1$. This favors balancing revision supervision with token-wise distillation rather than simply increasing the preference weight.
- Larger K weakens sustained performance, with accuracy at $K = 2$ falling from a 45.8% peak to 32.7% by the end of training. Expanding the revision budget can upset the balance between segment-wise preference learning and token-wise distillation, making training less stable.

6 Conclusion

In this work, we address the limitations of standard OPD, where teacher feedback on degenerate student prefixes may reinforce problematic reasoning patterns without explicitly providing coherent alternative reasoning steps. Our preliminary analysis shows that replacing student segments with teacher redrafts can improve subsequent reasoning accuracy. Building on these findings, we develop an empirically grounded method for learning reasoning revision in OPD, with a learning objective that encourages students to prefer teacher redrafts over their paired segments. Extensive experiments show that our SEG-OPD consistently outperforms the competitive baselines in average reasoning accuracy across diverse models and tasks, while achieving higher revision success rates and fewer reasoning loops than baselines. Ultimately, SEG-OPD bridges the gap between teacher guidance on student segments and students' capacity for reasoning revision, enabling them to redirect subsequent reasoning when their current reasoning risks becoming problematic.

AI USE STATEMENT

In this work, we used generative AI tools to illustrate the comic hiking metaphor in Fig. 1 and to help identify and assess reasoning steps from the model's output. Beyond the uses disclosed above, we did not use generative AI tools for any other tasks that required or recommended disclosure while completing this work. We have reviewed all AI-assisted work by double-checking the output and revising it as needed. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

In this work, we explore reasoning revision learning to improve OPD methods. We believe that our work conforms to the ICLR Code of Ethics. The proposed method, SEG-OPD, helps improve understanding of the effectiveness of learning reasoning revision and makes LLM post-training improvement more accessible. Despite the potential benefits, we acknowledge that more effective post-training techniques may also be misused to enhance or adapt LLMs for malicious purposes. We encourage responsible use of our method, together with the adoption of appropriate safeguards and ethical guidelines to mitigate potential risks.

REPRODUCIBILITY STATEMENT

In this work, we propose a novel approach to bridge the gap between teacher guidance on student rollouts and students' capacity for reasoning revision, enabling students to redirect subsequent reasoning when their current reasoning becomes problematic. To ensure reproducibility, we provide comprehensive details throughout the paper and supplementary materials. The methodology of our proposed SEG-OPD framework is described in detail in Section 4. Appendix A also provides a complete description of the implementation, including the datasets used, models trained, adaptations of baseline methods, all specific training parameters, and prompts we used in this work. Our code is also available in an anonymous repository. The link is provided at the end of the Abstract.

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.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Zhenyu Tan, Benyi Zhang, Guangzhong Sun, Guiquan Liu, and Junfeng Fang. On predictability of reinforcement learning dynamics for large language models. *arXiv preprint arXiv:2510.00553*, 2025.

Yuchen Cai, Ding Cao, Liang Lin, Chunxi Luo, Xin Xu, Kai Yang, Weijie Liu, Saiyong Yang, Tianxiang Zhao, Guangzhong Sun, et al. Learning to foresee: Unveiling the unlocking efficiency of on-policy distillation. *arXiv preprint arXiv:2605.11739*, 2026.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30377–30385, 2026.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL https://arxiv.org/abs/2502.01456.

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.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2212.07919.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 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 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, Ning Ding, and Zhiyuan Liu. Justrl: Scaling a 1.5b llm with a simple rl recipe, 2025a. URL https://arxiv.org/abs/2512.16649.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025b. URL https://arxiv.org/abs/2504.11456.

Zhilin Huang, Hang Gao, Ziqiang Dong, Yuan Chen, Yifeng Luo, Chujun Qin, Jingyi Wang, Yang Yang, and Guanjun Jiang. Learning from your own mistakes: Constructing learnable micro-reflective trajectories for self-distillation. *arXiv preprint arXiv:2606.18844*, 2026.

Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.07079.

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

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2206.14858.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.20050.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

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. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Junru Lu, Jiazheng Li, Siyu An, Meng Zhao, Yulan He, Di Yin, and Xing Sun. Eliminating biased length reliance of direct preference optimization via down-sampled KL divergence. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1047–1067. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.60. URL https://aclanthology.org/2024.emnlp-main.60/.

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. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=TC1jCxm4qH.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 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.

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

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. Offline reinforcement learning for LLM multi-step reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8881–8893. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-acl.464. URL https://aclanthology.org/2025.findings-acl.464/.

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

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=yfcpdY4gMP.

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

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

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

Haolei Xu, Xiaowen Xu, Haiwen Hong, Zixuan Ni, Hongxing Li, Yiwen Qiu, Weiming Lu, and Yongliang Shen. Pass the baton: Trajectory-relayed on-policy distillation. *arXiv preprint arXiv:2607.26057*, 2026b.

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*, 2025a. URL https://arxiv.org/abs/2505.09388.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin Cui, and Shuicheng Yan. SuperCorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations*, pp. 3386–3418, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0967d7c8b171dd81b77c43067c02bebf-Paper-Conference.pdf.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026. URL https://arxiv.org/abs/2602.12125.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 333–356, 2024. doi: 10.52202/079017-0011. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/00d80722b756de0166523a87805dd00f-Paper-Conference.pdf.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

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

Ziqi Zhao, Xinyu Ma, Liu Yang, Yujie Feng, Daiting Shi, Jingzhou He, Xin Xin, Zhaochun Ren, and Xiao-Ming Wu. Rosd: Reflective on-policy self-distillation for language model reasoning across domains. *arXiv preprint arXiv:2605.28014*, 2026b.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Mingyi Wang, Bo Peng, Jiayi Liu, Xiangjun Fan, and Zhuokai Zhao. OmniOPD: Logit-free on-policy distillation via speculative verification. *arXiv preprint arXiv:2606.01476*, 2026. URL https://arxiv.org/abs/2606.01476.

A Implementation Details

A.1 Details of RSR Evaluation

In the main text, we propose a metric, revision success rate (RSR), in two places: **(i)**. We first introduce this metric in Sec. 3.3 to compare student revision behavior under random and pivot anchors, and in **(ii)** Sec. 5.3 to compare student revision behavior during training with SEG-OPD and OPD.

In this subsection, we present details of RSR evaluation. Manual identification of reasoning steps can be time-consuming and subjective. To avoid this and apply a common evaluation procedure, we use GPT-5.6 Sol to identify reasoning steps in complete student rollouts, distinguish revision attempts from other steps, and assess whether each attempt corrects an earlier error with a substantively different alternative. This process serves solely in RSR evaluation and differs from SEG-OPD's boundary selection, since a reasoning step is not necessarily a multi-token span ended with a period, and vice versa. Here we provide a prompt template for identifying reasoning steps and assessing revision success.

RSR evaluation prompt

```
Given the problem and the student's complete response, identify all
reasoning steps in their original order.

Prefer complete sentences where possible, but this is not required.

Preserve the original wording.

[Revision assessment]
For each step, determine whether it attempts to correct a suspected
error in earlier reasoning.  For such an attempt, determine whether it
corrects an error in an earlier step by providing a substantively
different alternative.

Assess it in the context of the preceding reasoning.

Changes in wording or length alone do not count as successful
revisions.

[Output]
Return every extracted step with a yes/no judgment on whether it is a
revision attempt.  For each revision attempt, also provide a yes/no
judgment on whether it succeeds, with a brief but confirmative
explanation.

[Problem]
{problem}

[Student response]
{student_response}
```

Simply put, a revision attempt is a reasoning step that seeks to correct a suspected error in earlier reasoning, whether it succeeds or not. We compute RSR as the number of successful revision attempts divided by the total number of revision attempts, including both successful and unsuccessful attempts. Steps that do not attempt a revision are excluded from the denominator.

Additionally, we highlight that RSR evaluation and training segment construction serve different purposes by:

- **RSR evaluation** identifies steps by their reasoning content to assess revisions in complete student rollouts.
- **Training segment construction** uses construction boundaries around pivot anchors to construct local segments for teacher redrafting, without using the RSR evaluator to determine these boundaries.

The preference for complete sentences in evaluation does not require its step boundaries to coincide with those of the training segments.

A.2 Details of Revision Unit Construction

To provide coherent revision targets for teacher redrafting, we propose to extract complete sentences around reasoning pivots. Following the use of periods for step segmentation (Golovneva et al., 2023; Wang et al., 2025a), we use periods to identify sentence boundaries. For each pivot anchor $t_i \in T$ in a failed student rollout y, we search backward and forward for the surrounding periods and extract the enclosing sentence as $s_i = y_{s_i^b:s_i^e}$, where s_i^b and s_i^e index its first and last tokens. We score the candidates using Eq. (5) and greedily select non-overlapping segments in descending score order, up to the per-rollout budget K.

For each selected student segment s, we seek an alternative reasoning step under the same prefix q. We provide q and s to the fixed teacher, prompt it to replace s with its own reasoning, and let it continue until it solves the problem. From the resulting full draft $\tilde{r}$, we retain the leading sentence ending at the first period as the teacher redraft r only if $\tilde{r}$ passes the task-specific correctness check. Applying the same sentence boundary rule to both branches yields the revision unit $p = (q, s, r)$ for segment-wise preference learning. We send each prompt below as a single user message with thinking disabled.

Specifically, the prompt fields {`problem`} and {`q`} contain the problem and the preceding student response, respectively, which together form the conditioning prefix q; {`s_initial`} contains student segment s. We generate full teacher drafts with temperature 0.2 and top-p 0.9, allowing up to $8{,}192$ new tokens within a $32{,}768$-token context window. We discard sentence candidates without a complete right boundary, as well as empty teacher drafts and drafts truncated at the generation limit.

Teacher redrafting prompt for mathematical reasoning

```
Continue the mathematical solution from the exact point shown below.

[Problem]
{problem}

[Solution prefix]
{q}

[Student next step to replace; do not quote it]
{s_initial}

[Task]
Write a mathematically correct continuation beginning at exactly the
same point as the student step.  Start directly with solution text:
do not discuss the prompt, the student, or your instructions.
Continue naturally until the problem is fully solved, including the
final answer.
```

Teacher redrafting prompt for programming

```
Continue the Python solution from the exact point shown below.

[Problem]
{problem}

[Solution prefix]
{q}

[Student next step to replace; do not quote it]
{s_initial}

[Task]
Write a corrected continuation beginning at exactly the same point and
programming subgoal as the student step.  Start directly with solution
text:  do not discuss the prompt, the student, or your instructions.
Continue naturally until the problem is fully solved, ending with one
complete executable Python code block.
```

To assess the quality of revision units, we conduct an audit with the following settings:

Table 3: Revision unit quality assessment.

Student Model	Audit results	Math (%)	Code (%)
Qwen3-4B	Redraft preferred	86.9	89.7
	Segment preferred	2.6	6.4
	Tie	10.5	3.9
Qwen3-8B	Redraft preferred	87.3	88.4
	Segment preferred	3.2	5.9
	Tie	9.5	5.7

Table 4: Student–teacher pairs. The algorithm column refers to teacher RL training algorithms.

Mathematical Reasoning		
Student Model	**Teacher Model**	**Algorithm**
DeepSeek-1.5B	JustRL-DeepSeek-1.5B (He et al., 2025a)	GRPO
Qwen3-4B	Qwen3-4B-Non-Thinking-RL-Math (Yang et al., 2026)	GRPO
Qwen3-8B	Qwen3-8B-DAPO (Cai et al., 2025)	DAPO
Qwen3-14B	Qwen3-14B-Math-RL (Cai et al., 2026)	DAPO
Competitive Programming		
Student Model	**Teacher Model**	**Algorithm**
DeepSeek-1.5B	JustRL-DeepSeek-1.5B (He et al., 2025a)	GRPO
Qwen3-4B	Qwen3-4B-Non-Thinking-RL-Code (Yang et al., 2026)	GRPO
Qwen3-8B	Qwen3-8B-Code-RL (Cai et al., 2025)	DAPO
Qwen3-14B	Qwen3-14B-Code-RL (Cai et al., 2026)	GRPO

- We randomly sample 1,024 revision units generated from a Qwen3-4B and Qwen3-8B student-teacher pair on mathematical reasoning and programming.
- We use the GPT-5.6 Sol model to evaluate the prefix q, the retained segments s, and the redraft r, presented anonymously in random order without revealing the sources.
- We assess whether s and r can be evaluated as local alternatives under q, then compare their correctness and progress toward a solution. Different valid reasoning strategies are allowed.
- We also label pairs as ties when evidence is insufficient to indicate a clear preference.

We record the results in Tab. 3. From the table, we have the following key observations:

- The evaluator prefers teacher redrafts in 86.9%–89.7% of audited units, compared with 2.6%–6.4% for student segments. This pattern holds for both Qwen3-4B and Qwen3-8B in mathematics and programming, supporting the training preference $r \succ s$ under the audit's criteria of correctness and progress toward a solution.
- We assess the preference from the retained segments under q, without access to the full teacher draft. These judgments support using short redrafts as local preference targets based on the retained text. Ties remain in the denominator, so the redraft preference rates cover all audited units, including those without a clear winner.

A.3 Details of Models

To highlight the scalability of our SEG-OPD and the generalizability of our findings, we conduct extensive experiments across models with diverse parameter scales, ranging from 1.5B to 14B parameters. Our student models include publicly available pre-trained and fine-tuned checkpoints, as well as models that we locally trained using the `verl` framework.

For all student models reported in the main text, the corresponding teacher model is consistently the RL-trained version of its base model. This setting aligns with (Cai et al., 2026), which emphasizes using the same model architecture for both teacher and student models to ensure a fair comparison

Table 5: The statistics of datasets in our work.

Evaluation Dataset	Size	Description
Training on DeepMath-103K (He et al., 2025b)		
AIME24 AIME25 AIME26 (Cai et al., 2026)	30	American Invitational Mathematics Examination from year 2024 to 2026.
MATH500 (Lightman et al., 2024)	500	MATH subset with varied difficulty
MINERVA (Lewkowycz et al., 2022)	272	Undergraduate math and science
Training on Eurus-RL-Code (Cui et al., 2025)		
Codeforces (Cui et al., 2025)	123	Competitive programming
TACO (Cui et al., 2025)	382	Diverse algorithmic problems
HumanEval+ (Liu et al., 2023)	164	Python functions with expanded tests

and stable training. Tab. 4 lists the student–teacher pairs for mathematical reasoning and competitive programming, together with the teachers' RL training algorithms and sources.

A.4 Details of Datasets

In this work, we evaluate mathematical reasoning and competitive programming. Following (Cai et al., 2026), we fully leverage domain-specialized capabilities by training the model on DeepMath-103K (He et al., 2025b) for mathematical reasoning and Eurus-RL-Code (Cui et al., 2025) for competitive programming. After filtering, the training datasets contain 57,046 DeepMath-103K problems and 25,276 Eurus-RL-Code problems, covering a wide range of topics and difficulty levels. We evaluate our SEG-OPD and competitors on extensive datasets that vary in source, scale, and difficulty to highlight the transferability of different methods. Table 5 summarizes the datasets used in our experiments.

A.5 Details of Compared Methods

In the main text, we compare SEG-OPD with representative and state-of-the-art OPD methods. For a fair comparison, we reproduce all baselines in the same `verl` framework, using the student–teacher pairs, training data, and evaluation protocol described above. We tune each method's hyperparameters based on student performance on a validation split held out from the training dataset and select its best validation configuration. This approach ensures that we evaluate each method under its best-performing conditions. For instance, G-OPD with Qwen3-4B achieves 61.2% on AIME25 under our experimental setting, compared with 56.1% best performance reported in the original paper (Yang et al., 2026). Here we present the methods and their implementations as follows:

- **OPD** (Agarwal et al., 2024; Yang et al., 2026): This foundational OPD method trains the student on its own generated trajectories using token-level feedback from the teacher. We reproduce the sampled-token reverse-KL objective, with teacher log-probabilities evaluated at every response token.
- **E-OPD** (Jin et al., 2026): E-OPD augments standard OPD with forward-KL supervision at positions where the teacher has high entropy, encouraging the student to retain multiple plausible continuations. We adapt the authors' released implementation to our training framework, preserving the teacher-entropy gate and the forward-KL term over the teacher's top-k tokens.
- **G-OPD** (Yang et al., 2026): G-OPD scales the implicit reward defined by the log-probability difference between the teacher and a reference model. We reproduce this method with a reward extrapolation coefficient of 1.25 and use a frozen copy of the initial student as the reference model.
- **OmniOPD** (Zhou et al., 2026): OmniOPD weights entropy-selected student chunks by their agreement with teacher continuations, using only text outputs from the teacher. We reproduce its Bayesian chunk weighting and KL regularization from the paper, using ROUGE-1 F1 (Lin, 2004) to measure agreement and the frozen initial student to regularize unaudited tokens.

A.6 Other Implementation Details

We implement SEG-OPD and all baselines based on `PyTorch` 2.10.0, `verl` 0.8.0 (Sheng et al., 2024), `vLLM` 0.18.0 (Kwon et al., 2023), and `Transformers` 5.3.0 (Wolf et al., 2020). All the experiments are conducted on 8 × NVIDIA H100 80 GB GPUs.

- **Optimizer:** AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay.
- **Learning rate:** 5×10^{-6} with cosine decay and without warmup.
- **Rollout batch size:** 256 for Qwen3-14B and 512 for the other students.
- **Student rollout sampling:** One rollout per problem, with temperature 1.0 and top-p 1.0, without top-k filtering.
- **Maximum prompt length:** $2,048$ tokens.
- **Student response limit:** $22,000$ tokens for mathematical reasoning and $30,000$ tokens for competitive programming.
- **Seeds:** We use 42 by default. For repeated runs, we use additional randomly selected seeds.
- **Segment budget:** $K = 1$ unless otherwise specified.
- **Preference margin scale:** $\beta = 0.01$.
- **Segment-wise loss weight:** For mathematical reasoning, λ is set as $\{0.75, 0.35, 0.33, 0.5\}$ for DeepSeek-1.5B, Qwen3-4B, Qwen3-8B, and Qwen3-14B, respectively. For competitive programming, λ is set as $\{0.67, 0.9, 0.67, 0.75\}$ for DeepSeek-1.5B, Qwen3-4B, Qwen3-8B, and Qwen3-14B, respectively.
- **Training steps:** 48.
- **Training time validation frequency:** Every 4 steps unless otherwise specified.

At each checkpoint, we generate 4 responses per problem with temperature 1.0 and top-p 1.0, averaging correctness first over responses and then over problems. For all methods, we report mean Pass@1 from the final checkpoint after all training steps as reasoning accuracy, unless otherwise specified, such as peak values that refer to the highest observed values of the stated metrics over the displayed checkpoints. We provide an example training command for SEG-OPD in the next box.

SEG-OPD training command

```
python3 -m verl.trainer.main_ppo \
  data.train_files="$TRAIN_FILE" \
  data.val_files="$VAL_FILES" \
  data.train_batch_size=512 \
  data.max_prompt_length=2048 \
  data.max_response_length=22000 \
  data.seed=42 \
  +data.apply_chat_template_kwargs.enable_thinking=False \
  actor_rollout_ref.model.path="$STUDENT_MODEL" \
  actor_rollout_ref.actor.optim.lr=5e-6 \
  actor_rollout_ref.actor.optim.lr_scheduler_type=cosine \
  actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0.0 \
  actor_rollout_ref.actor.ppo_mini_batch_size=128 \
  actor_rollout_ref.actor.ppo_epochs=1 \
  actor_rollout_ref.actor.policy_loss.seg_opd_enable=True \
  actor_rollout_ref.actor.policy_loss.seg_opd_token_loss_coef=1.0 \
  actor_rollout_ref.actor.policy_loss.seg_opd_segment_loss_coef=1.0 \
  actor_rollout_ref.actor.policy_loss.seg_opd_dpo_beta=0.01 \
  actor_rollout_ref.actor.policy_loss.seg_opd_dpo_neutral_gate_cap=False \
  actor_rollout_ref.rollout.n=1 \
  actor_rollout_ref.rollout.temperature=1.0 \
  actor_rollout_ref.rollout.top_p=1.0 \
  actor_rollout_ref.rollout.top_k=-1 \
  +trainer.seg_opd_teacher_model_path="$TEACHER_MODEL" \
  +trainer.seg_opd_max_segments_per_rollout=1 \
  +trainer.seg_opd_full_redraft_max_tokens=8192 \
  +trainer.seg_opd_redraft_temperature=0.2 \
  +trainer.seg_opd_redraft_top_p=0.9 \
  trainer.n_gpus_per_node=8 \
  trainer.nnodes=1 \
  trainer.test_freq=4 \
  trainer.total_training_steps=48
```

B Additional Experiments and Analysis

B.1 Additional Analysis on Boundary Construction Rules

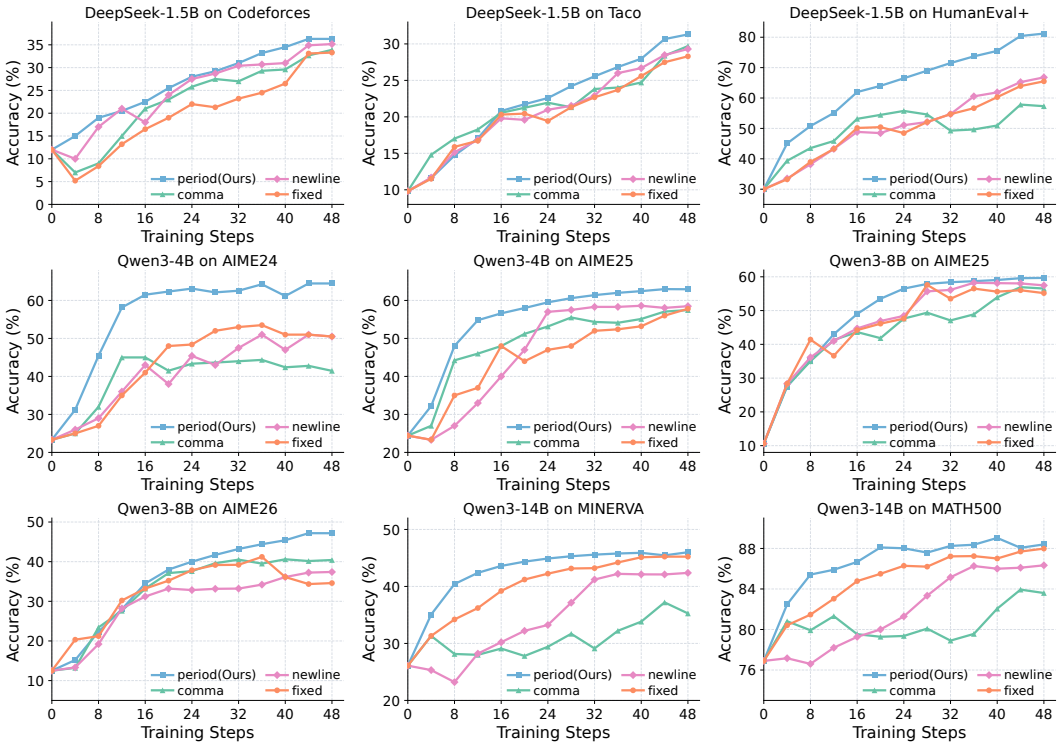


Figure 6: Reasoning accuracy of SEG-OPD with four different construction rules.

In the main text, we propose selecting boundaries from student rollouts to form segments. With selected reasoning pivots, SEG-OPD extends the boundaries from them in both directions until a full segment is formed. Primarily, we aim to build segments and redrafts that are both integrated and representative of the underlying reasoning process of the models, requiring us to preserve the completeness of the segment while ensuring it is informative. Punctuation and line breaks provide explicit structural cues in model outputs, allowing simple rules to guide both student segment construction and teacher redraft extraction. To achieve this, we conduct extensive analysis to assess the effectiveness of various construction rules in the context of our SEG-OPD. Specifically, we examine four different construction rules. We extend the boundaries from the selected pivots until we encounter the first **(i)** periods, **(ii)** commas, **(iii)** newlines, and **(iv)** a fixed length of $2,048$ tokens. We record the results for each construction rule across diverse models and datasets in Fig. 6. From the figure, we have the following observations:

- **Period** rule performs best overall, remaining at or near the top late in training. Period boundaries preserve sentence-level context for teacher redrafting. This helps retain a complete local inference and its explanation in the paired texts, making the reasoning contrast between student segments and teacher redrafts clearer for preference learning.
- **Comma** rule generally trails the period, with a clear gap on HumanEval+. Comma boundaries may fragment a reasoning step, separating an operation from its justification and obscuring the reasoning contrast between the student segment and teacher redraft.
- **Newline** rule approaches period accuracy on Codeforces but remains weaker on HumanEval+. Prior work uses double line breaks for step segmentation (Zhang et al., 2025), but line breaks can also reflect layout rather than reasoning boundaries.

- **Fixed** rule approaches the accuracy of that of the period rule on MINERVA but underperforms it on AIME26. Fixed token cuts may truncate a step or include later reasoning, especially when the student and teacher express the same step at different lengths.

B.2 Additional Analysis on Reasoning Behavior

To extend the analysis in Sec. 5.3, we examine how revision learning changes the student's use of reasoning transition markers and whether its RSR gains extend to additional model–dataset settings and a closely related baseline.

We report **word ratio**, the proportion of English words in natural-language reasoning that belong to our predefined set, including `wait`, `however`, `therefore`, `check`, `verify`, `correct`, `error`, `mistake`, and `incorrect`. Code is excluded from this calculation. Prior work uses related expressions as lexical indicators of reflection, backtracking, and self-verification (Huang et al., 2026). Motivated by this practice, we use word ratio as a lexical proxy for potential reasoning transitions and additionally report **RSR**, as defined in Sec. 3.3, to assess successful revisions.

In addition to OPD, we include OmniOPD (Zhou et al., 2026), a closely related baseline that also applies teacher supervision to multi-token chunks of student-generated trajectories. OmniOPD uses similarity with teacher continuations to weight learning from student chunks, whereas SEG-OPD directly learns to prefer teacher redrafts over paired student segments. We make the following observations from Fig. 7:

- SEG-OPD increases the use of transition markers across the displayed settings, with higher word ratios than both baselines late in training for DeepSeek-1.5B and Qwen3-14B. For DeepSeek-1.5B, the ratio approaches the initial teacher's level on both TACO and Codeforces. This reflects increased use of the selected reasoning-related expressions.
- Late in training, SEG-OPD attains higher RSR than OPD in all four additional settings, covering both mathematical and competitive programming. Successful revisions therefore account for a larger proportion of the student's revision attempts across these settings, extending the behavioral gains observed in the main text.
- Compared with OmniOPD, SEG-OPD generally reaches higher RSR earlier in training, while the gap later narrows on Qwen3-4B/AIME24 and Qwen3-8B/Codeforces. On Qwen3-8B/AIME26, a clear gap remains late in training. These trajectories show earlier RSR gains with SEG-OPD than with OmniOPD, even when their late-stage RSR becomes similar.

B.3 Additional Analysis on Training Dynamics of SEG-OPD

We investigate SEG-OPD's training dynamics by tracking its **(i)** loss gradient norms, **(ii)** training losses, and **(iii)** policy entropy. We compute gradient norms with respect to model parameters before clipping, and report component loss values before applying their mixture coefficients. Policy entropy measures the student's average uncertainty over the next token when generating responses.

Additionally, to examine the benefits of segment-wise preference learning over redraft imitation, we compare SEG-OPD with its SFT variant SEG-OPD $\setminus$N in Sec. 5.4. This variant replaces the segment-wise preference loss with SFT on teacher redrafts while retaining token-wise OPD. Specifically, we rewrite the segment-wise term $\mathcal{L}_s$ in Eq. (9) with a cross-entropy loss on teacher redrafts:

$$\mathcal{L}_s(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|r_i|}\sum_{t=1}^{|r_i|}\log \pi_\theta(r_{i,t} \mid q_i, r_{i,<t}), \tag{10}$$

where N is the number of revision units in the training batch and $|r_i|$ is the number of tokens in the retained teacher redraft r_i. The student policy π_θ, with parameters θ, predicts token $r_{i,t}$ given the prefix q_i and preceding redraft tokens $r_{i,<t}$. We first average the token losses within each redraft and then average across revision units, giving each unit equal weight in the loss. This ablation replaces segment-wise preference learning with direct teacher-redraft imitation while retaining token-wise OPD. The overall objective remains $\mathcal{L}_t + \lambda\mathcal{L}_s$, where $\mathcal{L}_t$ is the token-wise OPD loss, and λ controls the relative strength of redraft imitation. The SFT term omits the preference scale β and the com-

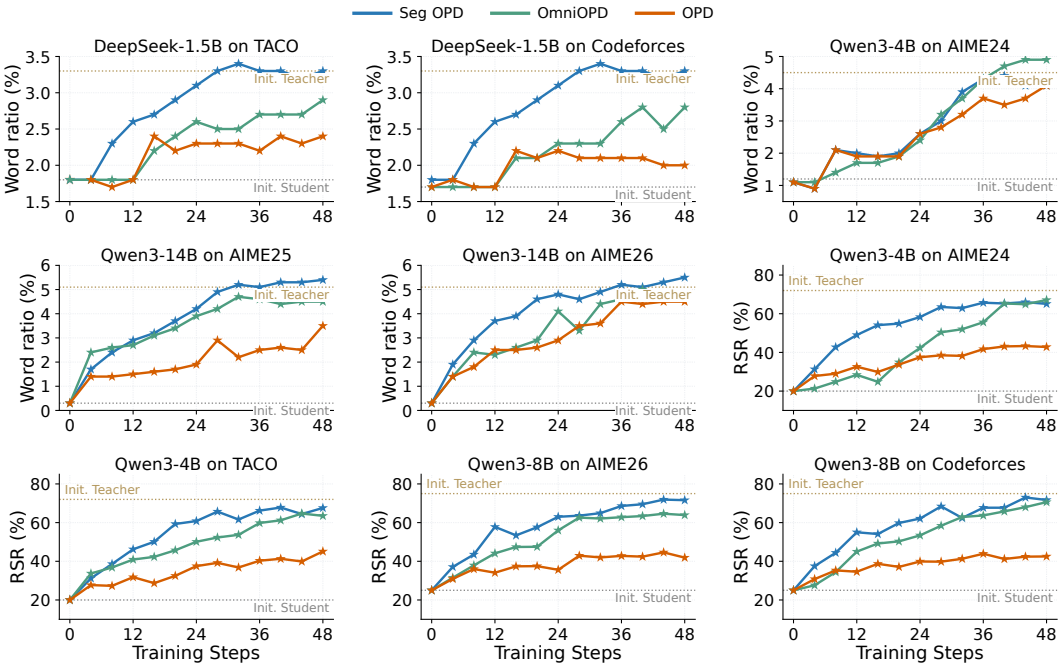


Figure 7: Additional analysis of reasoning behavior of SEG-OPD, OmniOPD, and OPD. The first five panels report word ratio, and the remaining four report revision success rate (RSR), ordered from left to right and top to bottom. Horizontal lines indicate the initial behavioral metric values for the student and teacher models, respectively.

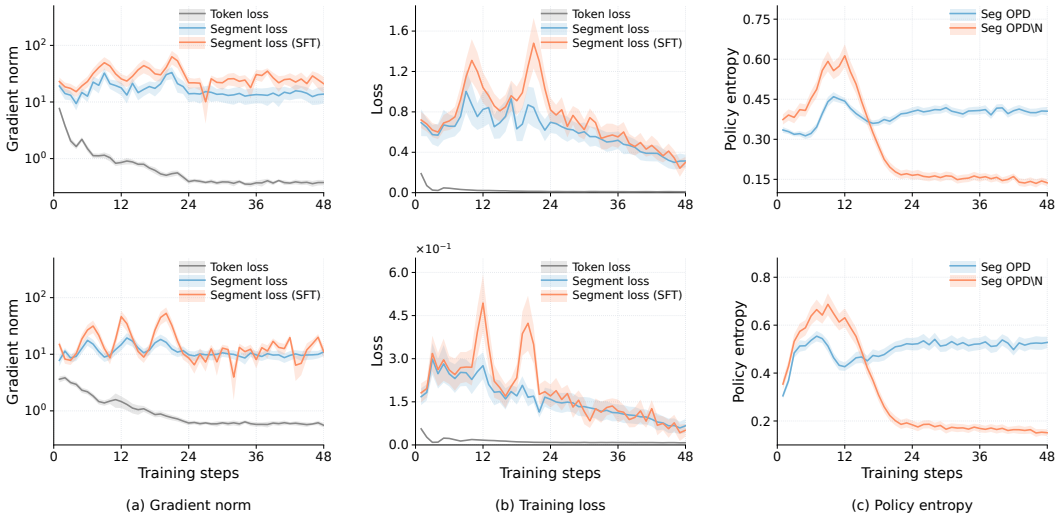


Figure 8: Training dynamics of training a Qwen3-4B on DeepMath-103K **(first row)** and Eurus-RL-Code **(second row)**. **Left** column and **middle** columns compare the gradient norms and values of the token loss, segment loss, and segment loss (SFT) used in SEG-OPD \N variant, respectively. **Right** column compares the policy entropy of SEG-OPD and its SFT variant. We record the mean and standard deviation across five runs with different random seeds.

parison against the student segment; thus, its gradient scale need not match that of the preference loss. The coefficient λ provides an overall rescaling of the SFT contribution.

We build on direct preference optimization (DPO), which learns from the contrast between preferred and rejected responses (Rafailov et al., 2023). In SEG-OPD, this contrast teaches the student to favor

a teacher redraft over the student segment it revises, rather than only increasing the redraft's likelihood. We normalize each segment's log policy ratio by its token length and rescale the difference by the mean segment length, as defined in Eq. (6).

Fig. 8 shows the training dynamics of Qwen3-4B on DeepMath-103K and Eurus-RL-Code. We make the following observations:

- **(i) Gradient norms.** The token-loss gradient norm generally decreases over training, while the segment-loss gradient norm continues to exhibit local fluctuations. Compared with SFT on teacher redrafts, the preference loss produces smaller gradient norms at most training steps, with lower peaks in both tasks. The larger SFT norms occur under per-redraft token averaging, not unnormalized token summation. Equal loss weight across revision units does not imply matched gradient scales between SFT and preference learning.
- **(ii) Training loss.** The token loss decreases in both tasks, indicating closer matching to the teacher's token predictions on sampled student trajectories. After early fluctuations, both the segment-wise preference loss and the SFT loss generally decrease. The preference loss penalizes small or negative revision margins, whereas the SFT loss measures how well the student predicts teacher redrafts. By explicitly favoring teacher redrafts over their paired student segments, SEG-OPD uses the contrast between two reasoning alternatives as a direct training signal. This teaches the student which local reasoning choice to prefer under the same prefix, complementing token-wise distillation with supervision tailored to its own reasoning. The accuracy gains over redraft SFT in Tab. 2 further support the value of learning from this contrast.
- **(iii) Policy entropy.** In both tasks, the SFT variant's policy entropy rises initially and then declines sharply, whereas SEG-OPD maintains higher entropy in later steps. Lower entropy corresponds to more concentrated next-token predictions on the sampled trajectories. The SFT variant also scores below SEG-OPD on all eight benchmarks in Tab. 2, with accuracy gaps ranging from 3.33% on AIME26 to 21.54% on AIME24. Together with the accuracy differences in Tab. 2, these observations support segment-wise preference learning over direct redraft imitation in this OPD setting.

B.4 CASE STUDY OF REASONING CORRECTION

In this subsection, we present two case studies of reasoning correction in student reasoning trajectories. Specifically, we organize these cases with the following settings:

- We present qualitative comparisons between the initial and SEG-OPD-trained student responses to the same mathematical reasoning and competitive programming problems, using saved evaluation rollouts.
- We highlight the same specific reasoning error in each response pair and show how the trained student subsequently corrects it.
- We report the final mathematical answers and the trained student's programming test results.

For the mathematical reasoning case, we select AIME26 Problem #4, which reduces a counting problem to finding distinct factors greater than one. The key distinction is between identifying an equal-factor decomposition and showing that no admissible distinct-factor decomposition exists. We focus on how the trained response checks a proposed exclusion rule: a correct algebraic reduction does not by itself justify excluding an entire class of numbers.

For the competitive programming case, we select TACO Problem #797C, where both responses share the stack representation. The key decision is when to pop the stack: once a character is appended to the output, later moves cannot change its position. We focus on how the response tests the proposed greedy rule against the given example and reconsiders which information from the unread input should guide that decision.

We present the detailed cases on the next pages.

A mathematical reasoning case

Question (AIME26, Problem #4)

Find the number of integers less than or equal to 100 that are equal to $a + b + ab$ for some choice of distinct positive integers a and b.

Initial Student Response

... But there's a clever way to simplify it: $a + b + ab = (a + 1)(b + 1) - 1$. ... Let's denote $x = a + 1$, $y = b + 1$. ... So the expression becomes: $n = xy - 1$. ...

Wrong reasoning step
We want to **exclude perfect squares** (since in that case $x = y$)

So first, let's list composite numbers ≤ 101, and then remove the perfect squares. ... So we remove these 9 perfect squares from the list of 53 composite numbers. ... So the total number of such n is: $\boxed{44}$.

SEG-OPD-trained Student Response

... Let $x = a + 1$, $y = b + 1$, so $x, y \geq 2$ (since $a, b \geq 1$), and $x \neq y$ (since $a \neq b$). Then the expression becomes: $N = xy - 1$. ... Let's denote $M = N + 1$. ...

Wrong reasoning step
So we need to count the number of integers $M \in [4, 101]$ such that M is composite and is **not a perfect square** (since perfect squares have only one distinct factorization into two equal integers)

Reasoning revision
Wait, not necessarily. For example, consider $M = 16$. It can be factored as 2×8, 4×4. So even though it's a perfect square, it still can be factored into distinct integers. So the perfect square condition is not sufficient to disqualify it.

... So the numbers that cannot be written as a product of two distinct integers $x, y \geq 2$ are:

- Those composite numbers which are perfect squares of primes, i.e., numbers of the form p^2 where p is prime

... So, of the 74 composite numbers from 4 to 101, 4 of them cannot be written as a product of two distinct integers ≥ 2. Therefore, the total number of integers $M \in [4, 101]$ that can be written as xy with $x, y \geq 2$, $x \neq y$, is: $74 - 4 = 70$. ... Final Answer: $\boxed{70}$.

We find that both responses initially exclude all perfect squares, but the SEG-OPD-trained student revises this rule after recognizing that $16 = 2 \times 8$ uses distinct factors. It then excludes only squares of primes from the composite numbers and obtains the correct answer of 70, whereas the initial student retains the error and answers 44.

A competitive programming case

Question (TACO, Problem #797C)

Petya received a gift: a string s of length up to 10^5 characters for his birthday. He took two more empty strings t and u and decided to play a game. This game has two possible moves:

- Extract the first character of s and append t with this character.
- Extract the last character of t and append u with this character.

Petya wants to make strings s and t empty and string u lexicographically minimal.

Input: `acdb` Output: `abdc`

Initial Student Response

To solve this problem, we need to simulate the game Petya is playing. The goal is to make the string `u` lexicographically minimal. ... Use a stack (`t`) to simulate the intermediate string.

Wrong reasoning step
At each step, check if the last character of the stack is less than or equal to the **next character** in `s`. If so, pop it and add it to `u`.

Otherwise, push the next character from `s` to the stack. At the end, pop all remaining characters from the stack to `u`. ...

SEG-OPD-trained Student Response

To solve this problem, we need to simulate the game that Petya is playing. The goal is to have the final string `u` as lexicographically minimal as possible. ... We can simulate the game by using a stack and a greedy approach.

Wrong reasoning step
We push characters from `s` onto the stack, and when the top of the stack is less than or equal to the **next character** in `s`, we pop it to `u`.

... Wait, but this is not correct for the example "`acdb`". ... `u` is `['a', 'c', 'b', 'd']` $\rightarrow$ but the correct answer is '`abdc`'. So this algorithm is not correct.

Reasoning revision
The correct solution is to compare the current top of the stack to the **minimum remaining character** in the remaining string. ... We can precompute the minimum suffix for each position.

... Now, for each position `i`, after pushing `s[i]` to the stack, we can check if the top of the stack is less than or equal to `min_suffix[i+1]`. If so, it's safe to pop it to `u`. ...

```
for i in range(n):
    t.append(s[i])
    while t and (i == n-1 or t[-1] <= min_suffix[i+1]):
        u.append(t.pop())

...
```

We find that both students initially compare the stack top only with the next input character, overlooking smaller characters that appear later. The SEG-OPD-trained student detects this error on `acdb` and revises the comparison to use the smallest character in the remaining string. Its final program returns the correct `abdc` instead of `acbd` and passes all validation tests.

C Training Algorithm

Algorithm 1 Algorithm of SEG-OPD

Require: π_θ, π_*, π_{ref}, x, λ, K.
Ensure: SEG-OPD-trained student model π_{θ^+}.
1: Initialize θ for student model π_θ and base reference policy π_{ref}, respectively;
2: Select student segments s from on-policy rollout y for x based on Eq. (5);
3: **while** not converged yet **do**
4: **for** $t = 1, 2, \ldots, N_{\text{step}}$ **do**
5: Construct accepted revision units $p = (q, s, r)$ through teacher redrafting;
6: Compute revision margins $m_{\pi_\theta}(p)$ based on Eq. (6);
7: Compute $\mathcal{L}_s$ and $\mathcal{L}_t$ based on Eqs. (7) and (8);
8: Update θ by optimizing Eq. (9);
9: **end for**
10: Resample on-policy rollout y and select s based on Eqs. (4) and (5);
11: **end while**
12: **return** π_{θ^+};

We formalize the training algorithm of SEG-OPD in Algorithm 1. In line 2, we first select up to K student segments s from the on-policy rollout y using the uncertainty scores in Eq. (5). From lines 3 to 11, we train the student by optimizing the joint objective in Eq. (9), where N_{step} denotes the number of training steps per iteration. In detail, we construct accepted revision units $p = (q, s, r)$ through teacher redrafting (line 5), calculate their revision margins using Eq. (6) (line 6), and compute the segment-wise preference and token-wise OPD losses using Eqs. (7) and (8) (line 7). We then update the student parameters θ with these losses (line 8) and resample the on-policy rollout to select new student segments (line 10). The teacher and reference policies remain fixed. When no revision unit is accepted, we set $\mathcal{L}_s = 0$ and retain all valid rollout tokens for token-wise OPD. Next, in Appendix D, we analyze the training algorithm's time complexity.

D Complexity Analysis

Table 6: Relative training time on DeepMath-103K, normalized to OPD within student–teacher pair.

Student	Teacher	OPD	OmniOPD	SEG-OPD
DeepSeek-1.5B	JustRL-DeepSeek-1.5B	1×	3.67×	1.89×
Qwen3-4B	Qwen3-4B-Non-Thinking-RL-Math	1×	3.89×	1.93×
Qwen3-8B	Qwen3-8B-DAPO	1×	3.41×	1.73×
Qwen3-14B	Qwen3-14B-Math-RL	1×	3.39×	1.87×

In this work, we propose SEG-OPD, a segment-wise OPD method that enables students to learn reasoning revision from explicit teacher redrafting. For an asymptotic complexity analysis, we treat model sizes, batch size, and the number of optimization passes as fixed and assume dense attention with KV-cached decoding.

Let L be the total length of each problem prompt and its student rollout in an iteration. Student generation, teacher scoring, and student optimization in OPD (Agarwal et al., 2024) together cost $O(L(1 + L))$ per iteration. OmniOPD (Zhou et al., 2026) samples n teacher continuations of at most c tokens at each of K selected chunks, giving a conservative total bound of $O(L(1 + L) + Kn(L+c)(1+L+c))$, including student optimization and reference-policy regularization. For SEG-OPD, let D bound the number of new tokens in each full teacher draft before boundary alignment; generating up to K drafts and learning from the revision pairs gives a total bound of $O(L(1 + L) + K(L+D)(1+L+D))$. These bounds include teacher-prefix prefill and reference-policy evaluation; boundary extraction adds only a linear scan of the generated text. Here, K denotes the per-rollout chunk or segment budget for the respective method; OmniOPD uses $K = 10$, $n = 10$, and $c = 50$, whereas SEG-OPD uses $K = 1$ and redrafts only selected failed rollouts. For fixed sampling

budgets and $D = O(L)$, all three methods have quadratic upper bounds on sequence length, but differ in the amount of teacher generation and optimization work.

For practical cost, SEG-OPD generates a complete teacher continuation from the selected prefix before extracting the leading sentence as r, which is detailed in Appendix A.2. Its generation cost therefore depends on the full draft rather than the retained redraft length; the subsequent extraction is negligible compared with model inference. The additional generation work depends on how many rollouts are redrafted and on the full draft lengths produced from their selected prefixes. Prefix processing and revision-pair optimization also contribute to training time. OmniOPD instead generates up to $K \cdot n \cdot c = 5{,}000$ teacher tokens across $K \cdot n = 100$ continuations per rollout, and processes their conditioning prefixes. We therefore report measured training time to quantify the practical cost of the additional revision supervision, including teacher generation and student optimization. We compare with OPD as the token-wise baseline and with OmniOPD, which also generates teacher text for selected student chunks.

Table 6 reports the measured training-time ratios for the mathematical reasoning runs under the implementation settings in Appendix A.6, normalized to OPD for each student–teacher pair. SEG-OPD timings include generating the full teacher drafts before extraction. Across the four student–teacher pairs, Seg-OPD requires less measured training time than OmniOPD: 1.73–$1.93\times$ the OPD training time, versus 3.39–$3.89\times$ for OmniOPD. Thus, SEG-OPD's measured training time remains below twice that of OPD in these settings, including full teacher draft generation and preference learning.

E DISCUSSIONS AND FUTURE WORK

In this work, we propose SEG-OPD, a segment-wise OPD method that enables students to learn reasoning revision from explicit teacher redrafting. We conduct extensive empirical analysis across diverse model architectures and downstream tasks, demonstrating our method's effectiveness in learning reasoning revision and improving student reasoning performance. As with other post-training methods, SEG-OPD may be misused to adapt LLMs for malicious purposes. We encourage responsible use and appropriate safeguards to mitigate these risks.

Several promising directions emerge naturally for future research, such as extending SEG-OPD to multi-teacher settings, including those with black-box teachers. The segment-wise objective relies on preference supervision rather than teacher logits. This would allow multiple teachers, including black-box models, to provide teacher redrafts for student segments, while a single white-box teacher supplies the logits required for token-wise OPD. We believe our methodology can provide insights into exploring these directions simultaneously.