ON THE OFF-POLICY TEACHER IN ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) has recently emerged as a promising post-training paradigm in which the student learns from trajectories generated by its own policy under dense teacher supervision. However, OPD introduces a fundamental asymmetry: although the sampled trajectories are on-policy for the student, they are off-policy for the teacher. The teacher is typically optimized to continue from prefixes generated by its own policy, but during OPD it must instead supervise prefixes generated by the student. Empirically, we find that its continuation performance degrades as these prefixes grow longer. To address this issue, we propose Student-COnditioned Updates of the Teacher (`SCOUT`), a co-training framework that adapts the teacher to student-generated prefixes. Alongside standard OPD updates, `SCOUT` periodically optimizes the teacher's conditional ability using reinforcement learning with verifiable rewards, where the teacher generates continuations from student prefixes and learns from outcome rewards. Controlled experiments show that `SCOUT` improves the teacher's ability to continue from student-generated prefixes, supporting the intended mechanism of student-conditioned teacher adaptation. Across multiple teacher–student configurations, model scales, and reasoning domains, `SCOUT` also consistently improves the effectiveness of on-policy distillation.

1 INTRODUCTION

On-policy distillation (OPD) has emerged as a new method for Large Language Model (LLM) post-training. The student generates its own trajectories and a stronger teacher provides dense token-level supervision along the student-generated states (Agarwal et al., 2024; Gu et al., 2024). While on-policy trajectories benefit student training, OPD introduces a teacher-side distribution shift, where the teacher supervises prefixes sampled from the student-induced trajectory distribution rather than its own. As autoregressive generation proceeds, differences in intermediate decisions, reasoning patterns and errors, can accumulate, causing student-generated prefixes to become increasingly unlikely under the teacher policy. The teacher is therefore asked to provide next-token supervision in settings that it would rarely encounter during its own generation. Eventually, the teacher's token-level guidance may become less reliable given the accumulated mismatch, which in turn is harmful to OPD training (Liu et al., 2026). Similar studies also suggest that the effectiveness of OPD depends not only on the teacher's standalone capability, but also on teacher-student compatibility and on the states at which supervision is provided (Li et al., 2026; Zhou et al., 2026; Xing et al., 2026).

Recent work mitigates this issue by regulating the supervision obtained from a fixed teacher. Prefix-based methods restrict distillation to earlier portions of student trajectories, either to reduce the cost of generating full rollouts or to avoid later prefixes where teacher supervision may deteriorate (Zhang et al., 2026a; Zhou et al., 2026). Other approaches retain a larger portion of the trajectory but modulate the teacher signal according to measures of uncertainty or teacher-student discrepancy, for example by changing the divergence objective, masking outlier tokens, or restricting updates to regions where the supervision is considered reliable (Jin et al., 2026; Xing et al., 2026). Despite these differences, they share a common design choice: the teacher remains fixed, while the student update is adapted to selectively use its supervision. This raises a complementary question that has received less attention: rather than deciding when to trust a fixed teacher, **can we train the teacher itself to provide better supervision on student-generated prefixes?**

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.

In this work, we make the teacher itself adaptive to the student-induced distribution. We introduce **SCOUT** (**S**tudent-**CO**nditioned **U**pdates of the **T**eacher), which explicitly trains the teacher on student-generated prefixes, turning teacher-side distribution shift from a fixed property of OPD into an optimization target. The student first samples a complete trajectory, from which a prefix is used to condition the teacher. The teacher generates a continuation from this prefix and is optimized using an outcome reward. The student, in turn, is trained with dense OPD over its full trajectory using the evolving teacher as supervision. This creates two coupled learning processes: the student learns from dense feedback on its own trajectories, while the teacher learns to better handle prefixes produced by the student. As the teacher improves on student-generated prefixes, it provides stronger supervision for subsequent student updates.

We evaluate SCOUT on mathematical reasoning across three teacher-student pairs, spanning different model scales, teacher training configurations, model families and additionally test its generalization to code generation. Across the three math settings, SCOUT consistently outperforms standard OPD with a frozen teacher, improving average accuracy by 1.2–2.6 points. On code generation, SCOUT improves the average score from 56.6 to 59.7, providing evidence that the approach extends beyond mathematical reasoning. Our analyses characterize where these gains come from. During training, the teacher progressively improves its ability to recover from student-generated prefixes. We further test whether the final gains could simply come from additional teacher training. A control with the same teacher updates but without student-prefix conditioning does not match SCOUT, showing that adaptation to student-generated states is central to the improvement. Finally, we also show that sparse teacher updates are sufficient to realize the benefit of student-conditioned adaptation and SCOUT is complementary to other OPD approaches.

Contributions. Our contributions are threefold: (1) we identify the teacher-side distribution shift as an important challenge in OPD and introduce teacher adaptation as a complementary optimization axis to regulating teacher supervision; (2) we introduce **SCOUT**, a simple and intuitive way to optimize along this new axis by training the teacher on student-generated prefixes with outcome-based RL, while retaining standard OPD for the student; (3) we introduce a rigorous evaluation framework for SCOUT, spanning two tasks and multiple teacher–student pairs and use it to demonstrate that teacher adaptation generalizes across model scales, model families, and task domains.

2 PRELIMINARIES

On-Policy Distillation. Let $x \sim \mathcal{D}$ denote an input prompt sampled from a training distribution $\mathcal{D}$. Additionally, let π_θ and π_ϕ denote the student and teacher models, respectively. Knowledge distillation aims to improve the student π_θ by transferring knowledge from the teacher π_ϕ. Given a response $y = (y_1, \ldots, y_T)$, the student and teacher define next-token distributions $\pi_\theta(\cdot \mid x, y_{<t})$ and $\pi_\phi(\cdot \mid x, y_{<t})$, where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the response prefix preceding token y_t.

In off-policy distillation, training responses are typically generated by the teacher or drawn from a fixed dataset. Consequently, the student is trained on prefixes whose distribution may differ from the prefixes induced by its own autoregressive generation at inference time. This distribution mismatch is a form of exposure bias (Ranzato et al., 2016) and can limit the effectiveness of distillation.

On-policy distillation addresses this mismatch by collecting responses from the current student policy $y^S \sim \pi_\theta(\cdot \mid x)$. At each student-generated prefix $y^S_{<t}$, the teacher provides a next-token distribution $\pi_\phi(\cdot \mid x, y^S_{<t})$, yielding dense token-level supervision on states actually visited by the student. The standard OPD objective is

$$\mathcal{L}_{\mathrm{OPD}}(\theta; \phi) = \mathbb{E}_{x \sim \mathcal{D}, y^S \sim \pi_\theta(\cdot|x)} \left[\frac{1}{|y^S|} \sum_{t=1}^{|y^S|} D\left(\pi_\theta(\cdot \mid x, y^S_{<t}), \pi_\phi(\cdot \mid x, y^S_{<t})\right) \right],$$

where $D(\cdot, \cdot)$ denotes a measure of divergence between the student and teacher next-token distributions, such as forward or reverse Kullback–Leibler (KL) divergence or Jensen–Shannon divergence. Unlike off-policy distillation, OPD trains the student on prefixes induced by its current policy. It therefore reduces the mismatch between training and inference-time contexts while retaining the teacher's dense distributional supervision.

Group Relative Policy Optimization. Like OPD, GRPO (Shao et al., 2024) trains on responses sampled from the current policy, but uses sequence-level rewards instead of teacher token distributions. Given a context c, GRPO samples G responses from a policy and assigns each a reward $r_i = R(c, z_i)$, where $z_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid c), \quad i = 1, \ldots, G$. These rewards are normalized within the group to obtain a relative advantage $\widehat{A}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$. For each token, GRPO computes the importance ratio between the current and behavior policies, $\rho_{i,t}(\theta) = \frac{\pi_\theta(z_{i,t}|c, z_{i,<t})}{\pi_{\theta_{\text{old}}}(z_{i,t}|c, z_{i,<t})}$, and optimizes a clipped policy-gradient objective with optional KL regularization to a reference policy.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|z_i|}\sum_{t=1}^{|z_i|}\min\left(\rho_{i,t}(\theta)\widehat{A}_i, \text{clip}(\rho_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}_i\right) - \beta D_{\text{KL}}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right)\right].$$

3 SCOUT: Student-Conditioned Updates of the Teacher

3.1 The Off-Policy Teacher in OPD

OPD enables on-policy student training by optimizing the student on its own generated trajectories, while the teacher remains off-policy, providing supervision on trajectories it did not generate. Specifically, the teacher is primarily optimized to model $\pi_\phi(\cdot \mid x, y^T_{<t})$, where the preceding context follows its own trajectory distribution. However, during OPD, it must provide supervision under $\pi_\phi(\cdot \mid x, y^S_{<t})$, conditioned on prefixes generated by the student. This discrepancy exposes the teacher to contexts that may be unlikely under its native generation distribution. As differences in intermediate reasoning accumulate, this mismatch may become more pronounced later in the trajectory. We therefore hypothesize that the teacher becomes less effective at providing supervision on longer student-generated prefixes. We refer to this phenomenon as the **off-policy teacher** issue.

To empirically examine the teacher's behavior under student-generated contexts, we evaluate its ability to continue from student prefixes $\pi_\phi(\cdot \mid x, y^S_{<t})$. For each question in AIME 2025, the student generates 4 complete responses, which we truncate at prefix ratios ranging from 10% to 90%. From each truncated prefix, we sample 4 independent teacher continuations and report their average final-answer accuracy. This measures how the teacher's continuation ability changes as it is conditioned on increasingly long student-generated prefixes. We use Qwen3-1.7B in non-thinking mode as the student and Qwen3-4B-Instruct-2507 as the teacher, with a maximum response length of 16,384 tokens for both, giving the teacher sufficient generation budget to recover from potentially suboptimal student prefixes.

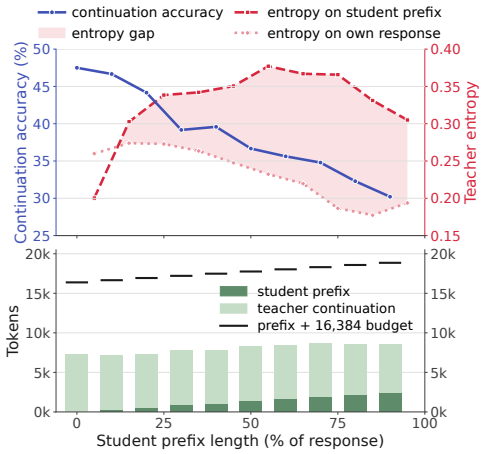

Figure 1: Teacher continuation from student-generated prefixes. As the student prefix grows longer, final-answer accuracy declines and a persistent entropy gap emerges between student and teacher prefixes, indicating increasing teacher uncertainty under student-generated contexts.

As shown by the blue line in Figure 1, teacher continuation accuracy generally decreases as the student prefix becomes longer. Importantly, the token-length statistics shown by the green bars indicate that the generated continuations remain well within the available response budget, ruling out insufficient generation length as the cause of this degradation. The decline therefore reflects a genuine deterioration in the teacher's ability to continue from $y^S_{<t}$.

We further compare the teacher's average next-token entropy when conditioned on its own prefixes, $\pi_\phi(\cdot \mid x, y^T_{<t})$, versus student prefixes, $\pi_\phi(\cdot \mid x, y^S_{<t})$, as shown by the red dashed lines in Figure 1. As the trajectory grows longer, entropy on teacher-generated prefixes decreases, whereas entropy on student-generated prefixes increases and then remains high, creating a persistent gap between

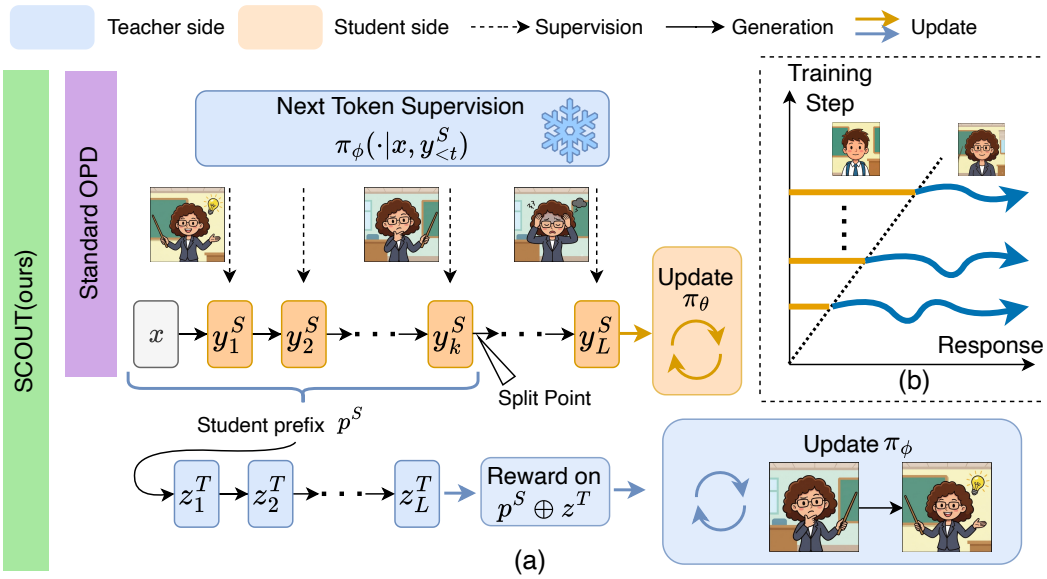

Figure 2: The overview of SCOUT and its comparison with standard OPD. While standard OPD suffers from unreliable teacher supervision at long prefixes, SCOUT periodically trains the teacher π_ϕ given the student prefix p^S. (a): SCOUT is a combination of standard OPD and teacher-side RL training. π_ϕ is synchronized after its update to provide supervision for the next OPD training steps. (b): π_ϕ becomes gradually better at student prefixes along training. So we linearly increase the student prefix ratio according to training steps to increase the difficulty level.

the two. Together with the decline in continuation accuracy, this shows that the teacher has greater difficulty operating on longer student-generated contexts.

3.2 METHOD

SCOUT retains the student-side training of standard OPD and augments it with periodic teacher adaptation, as illustrated in Figure 2(a). In standard OPD, the student generates a trajectory $y^S = (y^S_1, \ldots, y^S_T) \sim \pi_\theta(\cdot \mid x)$, and the teacher provides dense next-token supervision along this student-generated trajectory. SCOUT leaves this OPD update unchanged and adds a teacher-side RL update conditioned on prefixes from the same student trajectories as shown in the lower part of Figure 2(a). Specifically, we select a split point k and define the student prefix $p^S = y^S_{1:k}$. Conditioned on the prompt and p^S, the teacher samples K continuations,

$$z^T_i \sim \pi_\phi(\cdot \mid x, p^S), \qquad i = 1, \ldots, K. \tag{1}$$

Each hybrid response $p^S \oplus z^T_i$ receives a verifiable outcome reward, and the teacher is updated with RL using these rewards. Gradients are applied only to the teacher-generated continuation tokens. The updated teacher is then synchronized back to provide supervision for subsequent OPD updates. This yields two coupled learning processes: the student learns from dense teacher supervision on its own trajectories, while the teacher learns from outcome feedback on continuations of student-generated prefixes.

Since the student policy π_θ evolves throughout training, the distribution of student-generated contexts evolves with it. The teacher must therefore be updated periodically to remain effective on the contexts visited by the current student. For computational efficiency, we update π_ϕ once every f OPD steps, where f represents the update interval. We further increase the student-prefix ratio linearly over training, as illustrated in Figure 2(b). As observed in Section 3.1, longer student prefixes are more difficult for the teacher to continue from. We therefore begin teacher adaptation with shorter prefixes and gradually expose the teacher to longer and more challenging portions of the student trajectory as training progresses.

Table 1: Main results for Qwen3-4B-Instruct-2507 → Qwen3-1.7B. SCOUT consistently outperforms competing OPD methods across mathematical reasoning and code generation, demonstrating that its gains generalize across domains.

Method	Mathematical Reasoning							Code Generation			
	AIME24 *Avg@32*	AIME25 *Avg@32*	AMC23 *Avg@32*	HMMT *Avg@32*	Olym. *Avg@4*	MATH500 *Avg@8*	**Mean**	LCB v5 *Avg@4*	HE+ *Avg@16*	MBPP *Avg@8*	**Mean**
Qwen3-1.7B	13.3±4.2	10.3±3.2	46.8±4.9	5.4±2.6	42.3±1.5	73.1±1.5	31.9	27.3±0.3	62.4±1.5	47.3±1.2	45.7
Qwen3-4B-Ins	54.9±4.6	44.5±4.8	92.4±3.0	28.0±4.5	72.0±1.6	93.5±0.7	64.2	50.3±0.3	84.3±1.3	79.4±1.2	71.3
GRPO	30.9±4.5	30.4±4.1	73.3±5.7	18.9±3.4	61.1±5.8	87.4±3.5	50.3±4.4	44.3±6.2	72.7±1.4	69.6±5.0	62.2±2.8
OPD	33.8±3.0	24.3±1.8	73.9±1.9	16.0±0.8	60.2±1.6	87.0±1.1	49.2±1.5	37.6±0.1	69.7±0.7	62.5±0.3	56.6±0.3
ESR	31.9±0.9	25.5±1.1	70.6±2.6	16.0±1.4	59.0±0.5	86.3±0.6	48.2±0.5	36.0±2.3	68.1±1.0	61.3±2.2	55.1±1.7
Prune-OPD	34.0±0.8	27.3±0.8	69.8±0.7	16.5±0.8	58.7±0.8	85.9±0.2	48.7±0.2	36.8±0.2	69.4±0.7	61.2±0.4	55.8±0.4
Relay-OPD	**35.8±2.8**	24.8±1.2	73.9±1.2	15.0±1.5	59.9±0.2	86.3±0.2	49.3±0.6	2.3±0.4	14.3±1.6	19.6±0.9	12.1
SCOUT	**35.8**±0.5	**28.8**±0.6	**75.8**±0.6	**17.2**±1.9	**62.4**±0.8	**88.4**±0.3	**51.4**±0.6	**40.6**±2.5	**73.4**±0.9	**65.2**±2.6	**59.7**±1.9

Table 2: Mathematical reasoning with Qwen3-8B-DAPO → Qwen3-1.7B. SCOUT outperforms all other OPD-based methods across all six benchmarks, demonstrating that its gains generalize to larger model scales.

Method	Mathematical Reasoning						
	AIME24 *Avg@32*	AIME25 *Avg@32*	AMC23 *Avg@32*	HMMT *Avg@32*	Olym. *Avg@4*	MATH500 *Avg@8*	**Mean**
Qwen3-1.7B	13.3±4.2	10.3±3.2	46.8±4.9	5.4±2.6	42.3±1.5	73.1±1.5	31.9
Qwen3-8B-DAPO	59.9±6.0	43.9±5.9	90.7±3.9	25.7±4.1	74.2±0.7	94.6±0.5	64.8
GRPO	30.9±4.5	30.4±4.1	73.3±5.7	18.9±3.4	61.1±5.8	87.4±3.5	50.3±4.4
OPD	33.4±0.8	27.8±1.0	70.1±0.7	15.9±1.4	60.0±1.1	87.0±0.5	49.0±0.6
ESR	34.8±2.5	26.7±1.9	69.5±2.0	15.8±0.2	60.6±0.4	86.8±0.2	49.0±0.4
Prune-OPD	33.6±1.3	26.2±0.6	69.6±0.2	16.5±1.4	58.9±0.6	86.4±0.4	48.5±0.2
Relay-OPD	31.2±0.2	27.3±1.2	70.4±1.1	15.4±1.1	59.5±0.7	87.2±0.3	48.5±0.4
SCOUT	**35.4**±3.9	**31.2**±0.7	**75.0**±0.6	**17.7**±1.2	**61.4**±0.3	**89.0**±0.7	**51.6**±1.0

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate SCOUT on mathematical reasoning and code generation domains across three model pairs to test its generalization across task domains and model families.

Model configurations and training data. For the mathematical reasoning task, we experiment with (1) Qwen3-8B-DAPO as the teacher, obtained by training Qwen3-8B with GRPO on DAPO-MATH (Yu et al., 2025) for three epochs[1], and Qwen3-1.7B as the student; (2) Qwen3-4B-Instruct-2507 as the teacher and Qwen3-1.7B as the student; and (3) Skywork-OR1-Math-7B (He et al., 2025) as the teacher with DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI, 2025) as the student. We use OpenR1-Math-46K-8192 (Yan et al., 2025) as the training dataset. For the code generation task, we use Qwen3-4B-Instruct-2507 as the teacher and Qwen3-1.7B as the student and the TACO-Verified-7.5K subset of the DeepCoder training data (Luo et al., 2025) as the training dataset.

Baselines. We compare **SCOUT** against standard **OPD** with a frozen teacher, outcome-based RL using **GRPO**, and several recent OPD variants. **ESR** (Zhou et al., 2026) restricts distillation to an early portion of the student trajectory to avoid supervision on increasingly off-policy prefixes. **Prune-OPD** (Yang et al., 2026) dynamically restricts supervision according to local teacher–student compatibility. **Relay-OPD** (Xu et al., 2026a) allows the teacher to intervene during rollout generation. ESR uses a truncation length of 4096 following Xu et al. (2026a)[2]. SCOUT uses the same student-side distillation objective as standard OPD, while additionally updating the teacher through GRPO on continuations conditioned on student-generated prefixes.

[1]Because Qwen3-8B and Qwen3-1.7B are distilled from the same larger model (Yang et al., 2025), we further train the 8B model to provide additional task-specific capability for distillation (Li et al., 2026).

[2]The original truncation length 100 yields worse performance than 4096, so we choose the stronger one.

Table 3: Mathematical reasoning with Skywork-OR1-Math-7B → DeepSeek-R1-Distill-Qwen-1.5B. GRPO collapses and Prune-OPD degrades substantially in this setting, while SCOUT remains robust and outperforms baselines, demonstrating that its gains generalize across model families.

Method	Mathematical Reasoning						
	AIME24 *Avg@32*	AIME25 *Avg@32*	AMC23 *Avg@32*	HMMT *Avg@32*	Olym. *Avg@4*	MATH500 *Avg@8*	**Mean**
DeepSeek-R1-Distill-Qwen-1.5B	28.1±4.7	23.8±4.7	70.6±4.8	13.9±4.8	52.9±1.4	83.2±0.8	45.4
Skywork-OR1-Math-7B	57.7±5.4	46.4±4.8	90.2±2.7	27.3±3.9	73.4±0.9	94.8±0.7	65.0
GRPO†	4.1±3.0	3.1±2.7	35.8±5.3	0.4±1.1	29.2±1.4	63.1±1.2	22.6
OPD	37.8±0.6	**30.0±0.5**	77.9±0.6	**19.1±0.1**	60.6±0.2	89.1±0.1	52.4±0.2
ESR	32.6±2.0	27.3±0.1	71.6±1.7	17.3±1.1	56.9±0.4	87.5±0.1	48.8±0.4
Prune-OPD	21.6±1.0	18.8±0.4	56.8±0.3	11.2±0.3	40.5±0.7	72.0±0.4	36.8±0.1
Relay-OPD	34.7±1.8	28.3±1.0	74.6±1.6	16.4±0.8	58.9±1.0	88.3±0.2	50.2±0.6
SCOUT	**39.8±0.7**	**30.0±0.5**	**81.1±0.9**	18.7±0.3	**61.8±0.3**	**90.0±0.0**	**53.6±0.2**

Evaluation. We evaluate on six math reasoning benchmarks: AIME 2024 (Zhang & Math-AI, 2024), AIME 2025 (Zhang & Math-AI, 2025), AMC 2023, HMMT February 2025 (Balunovic et al., 2026), OlympiadBench (He et al., 2024), and MATH-500 (Lightman et al., 2024). For code generation, we evaluate on LiveCodeBench v5 (Jain et al., 2025), HumanEval+ (Chen et al., 2021; Liu et al., 2023), and MBPP (Austin et al., 2021).

To rigorously evaluate our methods, we account for randomness in both training and evaluation through repeated experiments. Each method is trained independently with three random seeds, and each resulting model is evaluated 4–32 times per benchmark, following benchmark-specific evaluation best practices. We report the aggregate mean accuracy and standard deviation (±) in the main tables. **Bold** and underline denote the best and second-best results among OPD-based methods, respectively. Appendix C.3 and Appendix D provides full evaluation details, complete per-run results, and significance tests against the standard OPD baseline.

4.2 Main Results

SCOUT scales across teacher sizes. We first test whether the gains from teacher adaptation persist as the teacher scales. Table 1 (left) and Table 2 use Qwen3-4B-Instruct-2507 and Qwen3-8B-DAPO teachers, respectively. SCOUT achieves mean accuracies of 51.4 and 51.6, improving over OPD by 2.2 and 2.6 points. It is also best or tied for best among OPD-based methods across all six benchmarks in both settings. These results show that SCOUT remains effective across teacher scales and training configurations.

SCOUT generalizes across task domains. Next, we evaluate the same Qwen3-4B-Instruct-2507 → Qwen3-1.7B pair on code generation. As shown in Table 1, SCOUT improves over OPD by 2.2 points on mathematical reasoning and 3.1 points on code generation. Relay-OPD, in contrast, remains competitive on math but drops to 12.1 mean accuracy on code, where its generations frequently contain syntax errors. This contrast suggests that modifying student rollouts can be sensitive to the task domain, while adapting the teacher to student-generated prefixes transfers more reliably.

SCOUT generalizes across model families. Our first two settings use Qwen3 teacher–student pairs. We therefore ask whether the same gains hold under a different model family, using Skywork-OR1-Math-7B to teach DeepSeek-R1-Distill-Qwen-1.5B. This setting is also more challenging for several baselines: GRPO collapses during training, while Prune-OPD substantially underperforms standard OPD. In contrast, SCOUT achieves a mean accuracy of 53.6, improving over OPD by 1.2 points and outperforming competing OPD methods overall. The gains therefore extend beyond the Qwen3 model family, even when the teacher–student pairing and training dynamics change.

Across these experiments, a consistent pattern emerges: adapting the teacher to student-generated prefixes improves OPD across changes in teacher scale, training configuration, model family, and task domain. SCOUT achieves the strongest aggregate performance among competing OPD methods in all three mathematical reasoning settings and in code generation, while several alternatives

†GRPO collapses during training on DeepSeek-R1-Distill-Qwen-1.5B; similar instability has been reported with the same framework (verl, 2025).

degrade sharply in specific settings. Together, these results suggest that student-conditioned teacher adaptation provides a simple and broadly effective way to improve on-policy distillation without relying on task- or model-specific interventions.

5 Analysis

Next, we study how teacher adaptation changes the training dynamics. We focus on three questions: **(RQ1)** Does teacher adaptation improve performance on student-generated prefixes? **(RQ2)** Are the gains specific to adapting the teacher on student-generated prefixes? **(RQ3)** How frequently should the teacher adapt to the evolving student? Finally, we evaluate under similar computational cost and whether SCOUT complements alternative approaches on improving teacher supervision.

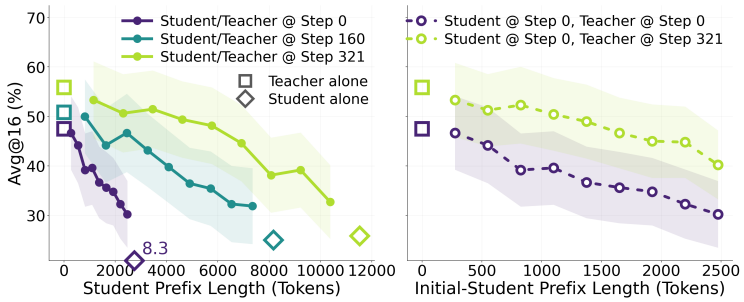

Figure 3: *SCOUT improves the teacher's ability to reason from student-generated prefixes.* **Left:** Matched student–teacher checkpoints become more accurate across increasingly long prefixes as training progresses. **Right:** Holding the student prefixes fixed, the trained teacher outperforms the initial teacher on the same inputs, showing that the gains reflect teacher adaptation rather than stronger student prefixes alone.

(RQ1) SCOUT improves the teacher's ability to continue from student-generated prefixes. We study how the teacher's ability to reason from student-generated prefixes changes during training in the Qwen3-4B-Instruct-2507 $\rightarrow$ Qwen3-1.7B math setting on AIME 2025. The student first generates a complete reasoning trajectory. We then select prefixes ending at different points in that trajectory and ask the teacher to continue reasoning from each prefix.

Figure 3 *(left)* compares matched student and teacher checkpoints at initialization, midway through training (Step 160), and after one epoch (Step 321). As training progresses, the continuation curves shift upward: later checkpoint pairs achieve higher accuracy across increasingly long student-generated prefixes. This shows that the student–teacher pair becomes increasingly effective at reasoning from longer student-generated prefixes. However, since later teachers are evaluated on prefixes from later, stronger students, this comparison alone does not tell whether the gains come from teacher adaptation or simply from better student-generated prefixes. Therefore, we hold the student prefixes fixed: Figure 3 *(right)* compares the teacher at initialization and after one epoch of training on prefixes generated by the *initial* student. The trained teacher achieves higher accuracy across all prefix lengths, confirming that the gains are not simply due to stronger student prefixes, but to improvement in the teacher itself.

Together, these results show the intended effect of SCOUT: as the student evolves during training, the teacher adapts to the prefixes it generates and becomes better at providing supervision on the student's own reasoning states.

(RQ2) SCOUT gains come from adapting the teacher to student-generated prefixes. Relative to standard OPD, SCOUT optimizes the teacher during training *and* conditions those updates on student-generated prefixes. This raises a natural question: do the gains come from additional teacher optimization, or from adapting the teacher specifically to student-generated states? To test this, we introduce **OPD + Teacher GRPO**. It trains the teacher with the same outcome-reward GRPO objective as SCOUT, but on rollouts that start from the original problem. In contrast, SCOUT starts these rollouts from prefixes generated by the student. This control tests whether additional teacher training alone is sufficient, without conditioning on student-generated states. Table 4 compares

Table 4: *Student-conditioned teacher adaptation drives the gains.* OPD + Teacher GRPO applies the same outcome-reward teacher updates as SCOUT, but starts teacher rollouts from the original problem rather than from student-generated prefixes.

Method	Qwen3-4B		Qwen3-8B
	Math	Code	Math
OPD	49.2	56.6	49.0
OPD + Teacher GRPO	48.8	57.0	50.8
SCOUT	**51.4**	**59.7**	**51.6**

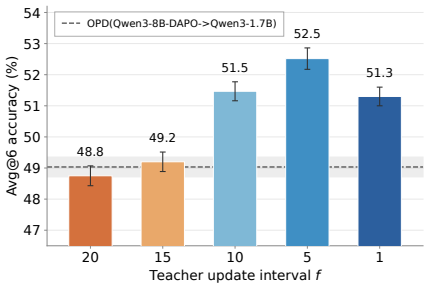

Figure 4: *Frequent teacher updates are unnecessary.* In the Qwen3-8B-DAPO → Qwen3-1.7B math setting, performance remains similar across moderate update intervals and drops only when updates become too sparse.

OPD, OPD + Teacher GRPO, and SCOUT across two teacher sizes, and on both math and code. Full per-benchmark results and individual training runs are reported in Tables 14 and 15.

Additional teacher training alone does not match the gains of SCOUT. On 4B math, OPD + Teacher GRPO slightly underperforms standard OPD, while on 4B code and 8B math it yields only modest improvements. In contrast, SCOUT achieves the strongest performance in all three settings. These results show that the gains do not come solely from additional teacher optimization; adapting the teacher to student-generated states is also important.

(RQ3) SCOUT does not require frequent teacher updates. We vary the teacher update interval f in the Qwen3-8B-DAPO → Qwen3-1.7B math setting to study how frequently the teacher must adapt as the student evolves. Figure 4 shows a broad range of effective update frequencies. Updating every 1, 5, or 10 student steps gives similar performance, with Avg@6 of 51.3, 52.5, and 51.5, respectively. However, when updates become too sparse (f=20), performance falls close to standard OPD.

These results show that the teacher must adapt to the evolving student, but does not need to follow every student update. We therefore use $f = 10$ in our main experiments: it preserves most of the performance gains while requiring fewer teacher updates. Full per-benchmark results are reported in Table 16.

Additional OPD training does not recover the gains of SCOUT. Unlike standard OPD, SCOUT spends additional compute on periodic GRPO updates to the teacher. We therefore test whether standard OPD can recover these gains simply by training longer on the same data. In the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting, we evaluate both methods every 40 training steps. We train SCOUT for one epoch and OPD for three, giving the two methods comparable total training compute.

Figure 5 shows that OPD improves initially but quickly plateaus, whereas SCOUT continues to improve as training compute increases. Since the additional OPD epochs reuse the same training data, this comparison does not rule out further gains from fresh data. Within this fixed-data setting, however, additional OPD training does not close the gap to SCOUT.



Figure 5: *Additional OPD training does not close the gap to SCOUT.* In the Qwen3-4B-Instruct-2507 → Qwen3-1.7B math setting, we train SCOUT for one epoch and OPD for three, on the same data, to compare performance over similar training-compute budgets. OPD quickly plateaus across repeated epochs, while SCOUT continues to improve.

SCOUT complements loss-level improvements to OPD. SCOUT adapts the teacher to student-generated prefixes, but this addresses only one source of unreliable supervision in OPD. Loss-level methods instead modify how the teacher signal is used during distillation. If these approaches address different failure modes, their gains should be complementary. We test this by combining

SCOUT with On-Policy Trust Region (which we shorten as OPTR), the first component of Trust Region On-Policy Distillation (TrOPD) (Xing et al., 2026). We evaluate OPD, OPTR, SCOUT, and OPTR+SCOUT in the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting.

OPTR improves over standard OPD on both math and code, and combining it with SCOUT yields a further 2.5-point gain on math and 1.2-point gain on code, reaching the strongest performance in both settings. This shows that teacher adaptation remains beneficial even with an improved distillation objective, suggesting that the two approaches address complementary aspects of teacher–student mismatch. Detailed per-benchmark and per-run results are reported in Table 17 and 18.

Table 5: *SCOUT provides complementary gains on top of OPTR.* In the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting, adding SCOUT improves both OPD and OPTR across math and code.

Task	Base	w/o SCOUT	+ SCOUT	Gain
Math	OPD	49.2	51.4	+2.2
	OPTR	49.8	**52.3**	+2.5
Code	OPD	56.6	59.7	+3.1
	OPTR	59.3	**60.5**	+1.2

6 Related Work

OPD trains the student on trajectories sampled from its own policy while using a stronger teacher for token-level supervision (Agarwal et al., 2024; Gu et al., 2024). As the student reasons, the teacher must supervise prefixes generated by the student rather than by its own policy, creating a teacher-side distribution shift. Much of recent work addresses this problem while keeping the teacher fixed and changing where or how its supervision is used.

One group of methods controls the rollout horizon. Prefix distillation, ESR, and TOPD stop supervision after a fixed early prefix (Zhang et al., 2026a; Zhou et al., 2026; Zhang et al., 2026b). POPD and Prune-OPD dynamically decide the prefix length (Zhang et al., 2026b; Yang et al., 2026).

Other methods retain more of the trajectory but change which teacher signals the student learns from. TIP, IW-OPD, SOD, and LGR reweight the loss or reward on tokens (Xu et al., 2026b; Xie et al., 2026; Zhong et al., 2026; Liu et al., 2026). TrOPD restricts OPD to regions where teacher supervision is considered reliable and treats outlier regions separately (Xing et al., 2026). There are also verifier-based methods that incorporate task outcomes into the supervision signal (Akhondzadeh et al., 2026; Lin et al., 2026; Qu et al., 2026).

A related line of work changes the trajectory or the form of corrective guidance. They generally detect the drift or early-failure in student trajectories and revise with the teacher, like MOTAB, Relay-OPD, TRD, and SOPD (Wang et al., 2026; Xu et al., 2026a; Jiang et al., 2026; Sun et al., 2026). TrOPD also includes an off-policy guidance component in which the student continues from teacher-generated prefixes (Xing et al., 2026).

Across these approaches, the teacher itself remains fixed. They improve supervision by changing which states are visited, which teacher signals are used, or how those signals are presented to the student. SCOUT takes a different but complementary direction: it updates the teacher so that it becomes better at supervising student-generated states. Teacher adaptation therefore provides an additional optimization axis for OPD.

7 Conclusions

We introduced SCOUT, a teacher–student co-training framework that addresses teacher-side distribution shift in on-policy distillation by adapting the teacher to student-generated prefixes. Across model pairs and task domains, SCOUT consistently improves over standard OPD and competing OPD methods. Our analysis reveals that the teacher becomes better at reasoning from student-generated prefixes, and that the gains come primarily from adapting it to the student rather than simply training the teacher further. Together, these results establish teacher adaptation as a promising and complementary optimization axis for improving on-policy distillation.

AI USE STATEMENT

In this work, we **used** generative AI tools for *providing feedback on research methodology or experiments*, *implementing methods*, *supporting qualitative data analysis*, and *interpreting results*.

We **have not used** generative AI tools to *help develop theoretical models or conceptual frameworks*, *propose or refine hypotheses*, and *clean and reformat datasets*.

Generating synthetic data sets, *formulating mathematical claims*, *providing critical ingredients for proving mathematical claims*, *assisting in the writing of proofs*, and *assisting with translation* are **not applicable** to this work.

Additionally, we **used** generative AI tools for *creating or modifying scientific figures or images*, *suggesting experimental parameters*, *drafting parts of a research paper*, *summarizing or analysing existing literature*, *discovering research topics or identifying gaps*, *brainstorming*, *searching for information*, *editing a research paper to improve readability*, *identifying relevant literature*, and *proposing a title or keyword for a research paper*.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We checked the correctness of the LLM-generated implementation, the qualitative data analysis, and results interpretations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies methods for improving on-policy distillation by adapting the teacher to student-generated contexts. The experiments use language models on mathematical reasoning and code-generation benchmarks and do not involve human subjects or the collection of personal or sensitive data.

Improving the efficiency and effectiveness of model distillation may make capable language models easier to train and deploy, but these improvements may also transfer to applications with potential for misuse. In particular, stronger reasoning and code-generation capabilities can support both beneficial and harmful downstream uses. Our experiments evaluate task performance rather than safety, robustness, fairness, privacy, or reliability, and the reported gains should therefore not be interpreted as evidence that the resulting models are suitable for safety-critical or consequential applications without additional evaluation and safeguards.

`SCOUT` also introduces additional computation by periodically updating the teacher during training. This increases training cost and associated energy use relative to fixed-teacher on-policy distillation, although the overall environmental impact depends on the hardware, workload, and energy source used. More broadly, we encourage future work on adaptive distillation methods to consider both their computational efficiency and the safety implications of transferring increasingly capable model behaviors.

REPRODUCIBILITY STATEMENT

We provide detailed information to facilitate reproduction of our results. Section 4.1 describes the model configurations, datasets, baselines, and evaluation benchmarks used in our experiments. Appendix C provides the training and inference hyperparameters, implementation details of the OPD objective, evaluation and statistical procedures, preparation of the Qwen3-8B-DAPO teacher, and the hardware and software environments used for training and evaluation. Appendix D reports complete per-benchmark and per-run results for the main experiments and ablations, including three independent training runs for all trainable methods.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=3zKtaqxLhW`.

Mohammad Sadegh Akhondzadeh, Vijay Lingam, Atula Tejaswi, Chanakya Ekbote, Sujay Sanghavi, and Aleksandar Bojchevski. Reward-gated on-policy distillation, 2026. URL https://arxiv.org/abs/2607.04037.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating LLMs on uncontaminated math competitions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=y0zL9IZxZ7.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report, 2025. URL https://arxiv.org/abs/2505.22312.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026. URL https://arxiv.org/abs/2606.08432.

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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=J5i09faOOf.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large

language models: Phenomenology, mechanism, and recipe, 2026. URL https://arxiv.org/abs/2604.13016.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Wenze Lin, Jiale Zhao, Xitai Jiang, Songde Rao, Yining Li, Shenzhi Wang, Bingxiang He, and Gao Huang. On-policy distillation with verifiable reward, 2026. URL https://arxiv.org/abs/2608.24696.

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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Yanjiang Liu, Jie Lou, Xinyan Guan, Yuqiu Ji, Hongyu Lin, Ben He, Xianpei Han, Le Sun, Xing Yu, and Yaojie Lu. Your teacher can't help you here: Combating supervision fidelity decay in on-policy distillation, 2026. URL https://arxiv.org/abs/2605.30833.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

Zikun Qu, Min Zhang, Mingze Kong, Zhiwei Shang, Zhengyu Chen, Yikun Ban, Shuang Qiu, and Zhongxiang Dai. Spot: Sparse probing and outcome calibration for on-policy distillation, 2026. URL https://arxiv.org/abs/2608.04419.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06732.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Changhui Sun, Lanbo Liu, Hang Lei, Tong Ling, Jiahang Xie, Zhiyong Zheng, Yujia Wang, Hao Liu, Feng Xiao, Lu Liu, Yanlong Du, Zifeng Cheng, Ziwei Jiang, and Qing Gu. Step-level on-policy distillation: Interpolating between on-policy distillation and supervised fine-tuning, 2026. URL https://arxiv.org/abs/2608.16333.

verl. Training collapse in GRPO. https://github.com/verl-project/verl/issues/2738, 2025. GitHub issue #2738.

Bing Wang, Shaotian Yan, Chen Shen, kaiyuan liu, Sinan Fan, Ximing Li, Rui Miao, Xiaosong Yuan, Zhanming Shen, and Jieping Ye. Backtracking when it strays: Mitigating dual exposure biases in llm reasoning distillation, 2026. URL https://arxiv.org/abs/2605.19433.

Yan Xie, Sijie Zhu, Tiansheng Wen, Bo Chen, and Yifei Wang. On the position bias of on-policy distillation, 2026. URL https://arxiv.org/abs/2606.22600.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation, 2026. URL https://arxiv.org/abs/2606.01249.

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, 2026a. URL https://arxiv.org/abs/2607.26057.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token importance in on-policy distillation, 2026b. URL https://arxiv.org/abs/2604.14084.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vO8LLoNWWk.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhicheng Yang, Zhijiang Guo, Yifan Song, Minrui Xu, Yongxin Wang, Yiwei Wang, Xiaodan Liang, and Jing Tang. Prune-OPD: Efficient and reliable on-policy distillation for long-horizon reasoning, 2026. URL https://arxiv.org/abs/2605.07804.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D Lyng, Sanjit Singh Batra, and Robert E. Tillman. Fast and effective on-policy distillation from reasoning prefixes. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25553–25569, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1276. URL https://aclanthology.org/2026.findings-acl.1276/.

Yaocheng Zhang, Jiajun Chai, Yuqian Fu, Songjun Tu, Xiaohan Wang, Wei Lin, Guojun Yin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Are full rollouts necessary for on-policy distillation?, 2026b. URL https://arxiv.org/abs/2605.31490.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024. URL https://huggingface.co/datasets/math-ai/aime24.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents, 2026. URL https://arxiv.org/abs/2605.07725.

Ziheng Zhou, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation, 2026. URL https://arxiv.org/abs/2605.27028.

APPENDIX CONTENTS

A **Limitations and Future Work** 15

B **Additional Discussions of the Related Work** 15

C **Experiment Details** 16

C.1 Hyperparameter Settings 16

C.2 Loss Implementation 16

C.3 Evaluation Details 17

C.4 Reproducibility and Compute 18

D **Complete Experiment Results** 18

D.1 Results of the Main Experiments 18

D.2 Results of the Analysis Experiments 19

A Limitations and Future Work

Following recent OPD work, our experiments focus on mathematical reasoning and function-level code generation. We broaden this evaluation across multiple teacher–student pairs, model families, and task domains. However, these benchmarks still represent relatively contained reasoning problems rather than long-horizon agentic tasks. Evaluating SCOUT on settings where trajectories extend over many decisions, such as repository-level software engineering or interactive agents, is therefore an important direction for future work. SCOUT also introduces additional cost by periodically generating teacher rollouts and updating the teacher with RL. Although our compute-matched analysis shows that simply training OPD longer on the same data does not recover the gains, SCOUT still incurs additional wall-clock and memory costs. Developing more efficient teacher updates is therefore an important next step. Moreover, our current teacher adaptation relies on verifiable outcome rewards. We therefore see learned or weaker feedback as a natural path toward extending SCOUT beyond verifiable domains. Finally, SCOUT assumes that the teacher model is directly accessible and trainable, and is thus not immediately applicable to black-box or API-based teachers whose parameters cannot be updated. One possible direction is to replace parameter updates with prompt-based teacher adaptation, potentially extending SCOUT to settings with inaccessible teachers.

B Additional Discussions of the Related Work

OPD trains the student on trajectories sampled from its own policy while using a stronger teacher for token-level supervision (Agarwal et al., 2024; Gu et al., 2024). As the student reasons, the teacher must supervise prefixes generated by the student rather than by its own policy, creating a teacher-side distribution shift. Much of recent work addresses this problem while keeping the teacher fixed and changing where or how its supervision is used.

One group of methods controls the rollout horizon. Prefix distillation and ESR stop supervision after an early prefix (Zhang et al., 2026a; Zhou et al., 2026). POPD gradually extends the rollout horizon during training, while TOPD uses a fixed truncated horizon (Zhang et al., 2026b). Prune-OPD monitors local teacher–student compatibility and, when the models drift apart, down-weights later supervision and truncates the rollout (Yang et al., 2026).

Other methods retain more of the trajectory but change which teacher signals the student learns from. IW-OPD reweights tokens using the accumulated teacher–student discrepancy (Xie et al., 2026), while SOD adjusts the distillation weight at each reasoning step using step-level divergence (Zhong et al., 2026). TIP identifies informative tokens using student entropy and teacher–student divergence (Xu et al., 2026b). TrOPD restricts standard OPD to regions where teacher supervision is considered reliable and treats outlier regions separately (Xing et al., 2026). LGR instead looks one step ahead, favoring student tokens that lead to higher teacher confidence at the next step (Liu et al., 2026). Verifier-based methods incorporate task outcomes into the supervision signal: RG-OPD uses verifier feedback to gate teacher supervision (Akhondzadeh et al., 2026), OPDVR uses trajectory correctness to gate token-level rewards (Lin et al., 2026), and SPOT uses verifier-scored continuations to construct outcome-calibrated distillation targets (Qu et al., 2026).

A related line of work changes the trajectory or the form of corrective guidance. MOTAB detects student drift, backtracks to an earlier safe state, and uses the teacher to redirect generation (Wang et al., 2026). Relay-OPD lets the teacher briefly take over at detected failure points before returning generation to the student (Xu et al., 2026a). TRD revises student rollouts under teacher guidance before distillation (Jiang et al., 2026). SOPD keeps student-visited prefixes but asks the teacher to generate a complete reasoning step from each prefix, providing longer-horizon supervision than token-level OPD (Sun et al., 2026). TrOPD also includes an off-policy guidance component in which the student continues from teacher-generated prefixes (Xing et al., 2026).

Across these approaches, the teacher itself remains fixed. They improve supervision by changing which states are visited, which teacher signals are used, or how those signals are presented to the student. SCOUT takes a different but complementary direction: it updates the teacher itself so that it becomes better at supervising student-generated states. Teacher adaptation therefore provides an additional optimization axis for OPD; our experiments further show that it can be combined with the loss-level OPTR component of TrOPD.

C Experiment Details

C.1 Hyperparameter Settings

We present the general hyperparameter settings in Tables 6 and 7. The performance varies with the choice of learning rate. Therefore, we conduct a grid search for the student model learning rate among $\{1, 3, 5, 7\} \times 10^{-6}$ on standard **OPD**. With the fixed student learning rate, we further search the teacher model learning rate among $\{1, 2, 3, 4, 5\} \times 10^{-6}$ for SCOUT. The best learning rates are shown in Table 8.

GRPO uses a learning rate of 3×10^{-6}. The rest of the hyperparameters are the same as OPD, except that its *Rollout Number per Prompt (n)* is 8. The reference weight $\beta = 0.001$.

In SCOUT, the update of the teacher model uses GRPO with the same settings as student-side GRPO described above, except for its learning rate shown in Table 8.

Table 6: Hyperparameters in Math reasoning task.

Hyperparameter	**Value**
Max Prompt Length	1024
Max Response Length	16384
Training Rollout Temperature	1.0
Training Rollout Top-p	1.0
Rollout Number per Prompt (n)	1
Global Batch Size	128
PPO Mini-Batch Size	128
PPO Epochs	1
PPO Clipping Range	0.2
Learning-Rate Schedule	constant
Training Epochs	1
Inference Temperature	0.6
Inference Top-p	0.95
Inference Max Response Length	16384

Table 7: Hyperparameters in Code generation task.

Hyperparameter	**Value**
Max Prompt Length	1536
Max Response Length	16384
Training Rollout Temperature	1.0
Training Rollout Top-p	1.0
Rollout Number per Prompt (n)	1
Global Batch Size	128
PPO Mini-Batch Size	128
PPO Epochs	1
PPO Clipping Range	0.2
Learning-Rate Schedule	constant
Training Epochs	3
Inference Temperature	0.6
Inference Top-p	0.95
Inference Max Response Length	16384

C.2 Loss Implementation

Although OPD is formulated using the reverse KL divergence, computing the exact full-vocabulary divergence requires evaluating the teacher distribution over the entire vocabulary at every student-generated token, which introduces substantial computational and memory overhead. We therefore adopt the policy-gradient formulation of OPD and use the single-sample K1 estimator. Specifically, for each student-generated token y_t, we define the token-level advantage as

$$A_t^{\mathrm{OPD}} = \mathrm{sg}[\log \pi_T(y_t \mid s_t) - \log \pi_{\theta_{\mathrm{old}}}(y_t \mid s_t)], \tag{2}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator and $\pi_{\theta_{\mathrm{old}}}$ is the student policy used to generate the rollout. The student is then optimized with the clipped policy-gradient objective

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = -\mathbb{E}_t \left[\min \left(r_t(\theta) A_t^{\mathrm{OPD}}, \mathrm{clip}(r_t(\theta), 1 - \epsilon_{\mathrm{low}}, 1 + \epsilon_{\mathrm{high}}) \, A_t^{\mathrm{OPD}} \right) \right], \tag{3}$$

where

$$r_t(\theta) = \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(y_t \mid s_t)} \tag{4}$$

is the importance-sampling ratio between the updated and rollout policies. This formulation avoids explicitly computing the full-vocabulary KL divergence while providing an efficient sampled approximation to the reverse-KL objective.

Table 8: Learning rates for the student and teacher models across different model pairs and tasks.

Teacher Model	Student Model	Task	Student LR	Teacher LR
Qwen3-4B-Instruct	Qwen3-1.7B	Math	5×10^{-6}	5×10^{-6}
Qwen3-8B-DAPO	Qwen3-1.7B	Math	3×10^{-6}	5×10^{-6}
Skywork-OR1-MATH-7B	DeepSeek-R1-Distill-Qwen-1.5B	Math	3×10^{-6}	5×10^{-6}
Qwen3-4B-Instruct	Qwen3-1.7B	Code	5×10^{-6}	5×10^{-6}

C.3 Evaluation Details

We adopt 6 benchmarks for the reasoning task and 3 benchmarks for the code generation task. Table 9 lists the number of questions for each benchmark and the number of repeated evaluations we conduct. We repeat 32 times for benchmarks with few questions: AIME 24, AIME 25, AMC, and HMMT. For HumanEval+, we evaluate them 16 times. For MATH500 and MBPP, we evaluate them 8 times. For OlympiadBench, we filter the text-only, automatically verifiable math questions, which results in 581 questions. For LiveCodeBench, we use the stable v5 version with 880 questions, and repeat evaluation 4 times.

For each OPD-based method, we conduct three independent training runs and evaluate every resulting checkpoint using the repeated-evaluation protocol described above. Tables 10, 11, 12, and 13 provide the detailed results. For an individual training run, we report the mean accuracy across repeated evaluations together with its standard error of the mean (SEM). In the aggregate rows, we report the mean across the three training runs together with the sample standard deviation (SD) of their run-level mean accuracies.

More precisely, let $y_{m,b,r}$ and $e_{m,b,r}$ denote the mean accuracy and within-evaluation SEM, respectively, for method m, benchmark b, and training run r. With $R = 3$ training runs, the aggregate accuracy and the reported cross-run SD are

$$\bar{y}_{m,b} = \frac{1}{R}\sum_{r=1}^{R} y_{m,b,r}, \qquad s^2_{m,b} = \frac{1}{R-1}\sum_{r=1}^{R}\left(y_{m,b,r} - \bar{y}_{m,b}\right)^2. \tag{5}$$

Thus, the aggregate table entry is $\bar{y}_{m,b} \pm s_{m,b}$. For significance testing on an individual benchmark, we use a two-level random-effects variance decomposition that separates variation across training runs from uncertainty due to repeated evaluation. We estimate the between-run variance as

$$\hat{\tau}^2_{m,b} = \max\left(0,\ s^2_{m,b} - \frac{1}{R}\sum_{r=1}^{R} e^2_{m,b,r}\right), \tag{6}$$

and estimate the variance of the aggregate mean by

$$V_{m,b} = \frac{\hat{\tau}^2_{m,b}}{R} + \frac{1}{R^2}\sum_{r=1}^{R} e^2_{m,b,r}. \tag{7}$$

For a method m and the corresponding OPD baseline 0 (Frozen-OPD in the Skywork setting), we test $H_0 : \mu_{m,b} \le \mu_{0,b}$ against $H_1 : \mu_{m,b} > \mu_{0,b}$ using

$$z = \frac{\bar{y}_{m,b} - \bar{y}_{0,b}}{\sqrt{V_{m,b} + V_{0,b}}}, \qquad p = 1 - \Phi(z), \tag{8}$$

where Φ is the standard normal cumulative distribution function.

For the Avg. column, we first compute the equal-weight macro-average within each training run,

$$M_{m,r} = \frac{1}{B}\sum_{b=1}^{B} y_{m,b,r}, \tag{9}$$

and then aggregate these run-level macro-averages. Here, $B = 6$ for the mathematical reasoning results, and $B = 3$ for code generation. We compare the three run-level macro-averages of each

Table 9: The statistics of the evaluation benchmarks, including the number of questions and the number of sampled responses per question.

Benchmark	Reasoning						Code Generation		
	AIME24	AIME25	AMC23	HMMT	Olympiad	MATH500	LCB v5	HumanEval+	MBPP
# Questions	30	30	40	30	581	500	880	164	500
# Repeats	32	32	32	32	4	8	4	16	8

method against those of the baseline using a one-sided Welch's t-test:

$$t = \frac{\bar{M}_m - \bar{M}_0}{\sqrt{s^2_{M,m}/R_m + s^2_{M,0}/R_0}}, \tag{10}$$

with Welch–Satterthwaite degrees of freedom

$$\nu = \frac{\left(s^2_{M,m}/R_m + s^2_{M,0}/R_0\right)^2}{\frac{\left(s^2_{M,m}/R_m\right)^2}{R_m - 1} + \frac{\left(s^2_{M,0}/R_0\right)^2}{R_0 - 1}}. \tag{11}$$

The one-sided p-value is $p = 1 - F_{t_\nu}(t)$. Orange cells indicate cases in which a method achieves a higher mean than the corresponding baseline and the one-sided test gives $p < 0.1$. The main experimental tables report the corresponding aggregate mean and cross-run SD over the three training runs.

C.4 Reproducibility and Compute

All experiments are run on a Slurm-managed cluster of AWS p4d.24xlarge instances, each equipped with 8 NVIDIA A100 40GB GPUs. All trainable methods are run with three independent seeds. We use no SFT warm-up stage: training starts directly from the released instruction or reasoning checkpoints.

For the Qwen3-8B-DAPO teacher used in our math experiments, we train Qwen3-8B with critic-free GRPO on the deduplicated DAPO-Math-17k dataset for exactly three epochs. We use a train and mini-batch size of 64, 8 rollouts per prompt, a constant learning rate of 1×10^{-6}, rollout temperature 1.0.

For math experiments, GRPO, Prune-OPD, and Relay-OPD use one node (8 GPUs), while frozen-teacher OPD and ESR use two nodes (16 GPUs). SCOUT uses three nodes (24 GPUs) with a 4B teacher and four nodes (32 GPUs) with a 7B/8B teacher. Code experiments use the same general resource allocation. In `SCOUT`, student and teacher training are placed on separate physical nodes using Ray resource pools. All evaluations are run on a single 8-GPU node with independent vLLM workers.

Our main software stack uses PyTorch 2.9.0, vLLM 0.12.0, Ray 2.55.1, verl 0.9.0.dev0, and Transformers 4.57.6. Training and inference hyperparameters are provided in the preceding appendix sections.

D Complete Experiment Results

This section provides the complete results corresponding to the experiments reported in the main paper. In addition to the aggregate results presented in the main text, we report per-benchmark performance and individual training runs for each experimental setting.

D.1 Results of the Main Experiments

For the main experiments, each method is trained with three independent runs, unless otherwise specified. For individual runs, the $\pm$ term denotes the standard error from repeated evaluation, while in the *Mean* rows it denotes the standard deviation across training runs. One-sided p-values for

Table 10: Complete results for the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting on six math reasoning benchmarks. This table corresponds to the mathematical reasoning task in Table 1. SCOUT shows significant improvement over OPD on 3 of 6 benchmarks and in the average performance.

Method	**Run** *Avg.@*	**AIME24** *32*	**AIME25** *32*	**AMC23** *32*	**HMMT** *32*	**Olymp.** *4*	**MATH500** *8*	**Avg6** –
GRPO	Run 1	29.06 ± 0.97	30.10 ± 0.61	73.28 ± 0.76	16.77 ± 0.69	60.80 ± 0.48	87.85 ± 0.11	49.64
	Run 2	27.71 ± 0.98	26.46 ± 0.70	67.58 ± 0.84	16.98 ± 0.48	55.46 ± 0.22	83.67 ± 0.42	46.31
	Run 3	36.04 ± 1.26	34.58 ± 0.87	79.06 ± 0.94	22.81 ± 0.54	67.00 ± 0.33	90.62 ± 0.26	55.02
	Mean	30.94 ± 4.47	30.38 ± 4.07	73.31 ± 5.74	18.85 ± 3.43	61.09 ± 5.78	87.38 ± 3.50	50.32 ± 4.39
OPD	Run 1	30.94 ± 1.06	22.40 ± 0.81	71.95 ± 0.71	15.42 ± 0.65	58.30 ± 0.45	85.75 ± 0.52	47.46
	Run 2	36.98 ± 0.97	25.83 ± 0.87	74.22 ± 0.82	15.83 ± 0.54	60.71 ± 0.31	87.95 ± 0.40	50.25
	Run 3	33.54 ± 1.00	24.69 ± 0.86	75.62 ± 0.84	16.88 ± 0.86	61.45 ± 0.60	87.40 ± 0.19	49.93
	Mean	33.82 ± 3.03	24.31 ± 1.75	73.93 ± 1.85	16.04 ± 0.75	60.15 ± 1.64	87.03 ± 1.14	49.21 ± 1.53
ESR	Run 1	31.56 ± 1.31	26.77 ± 0.71	71.09 ± 0.74	17.19 ± 0.43	58.48 ± 0.28	86.95 ± 0.15	48.67
	Run 2	31.25 ± 1.16	24.90 ± 0.78	67.81 ± 0.66	16.46 ± 0.67	59.34 ± 0.43	86.22 ± 0.25	47.66
	Run 3	32.92 ± 1.11	24.90 ± 0.97	72.89 ± 0.79	14.48 ± 0.71	59.25 ± 0.45	85.72 ± 0.26	48.36
	Mean	31.91 ± 0.89	25.52 ± 1.08	70.60 ± 2.58	16.04 ± 1.40	59.02 ± 0.47	86.30 ± 0.62	48.23 ± 0.52
	p vs. OPD	0.803	0.187	0.925	0.500	0.821	0.801	0.808
Prune-OPD	Run 1	33.44 ± 1.07	27.60 ± 0.78	68.98 ± 0.91	15.73 ± 0.74	59.51 ± 0.32	86.17 ± 0.39	48.57
	Run 2	33.65 ± 1.28	27.92 ± 0.77	70.00 ± 1.05	16.56 ± 0.63	57.87 ± 0.29	85.72 ± 0.59	48.62
	Run 3	34.90 ± 0.95	26.46 ± 0.81	70.39 ± 1.07	17.29 ± 0.62	58.61 ± 0.47	85.85 ± 0.36	48.92
	Mean	33.99 ± 0.79	27.33 ± 0.77	69.79 ± 0.73	16.53 ± 0.78	58.66 ± 0.82	85.91 ± 0.23	48.70 ± 0.19
	p vs. OPD	0.466	0.0391	0.980	0.242	0.871	0.887	0.689
Relay-OPD	Run 1	37.19 ± 0.96	23.54 ± 0.96	73.05 ± 0.94	13.23 ± 0.59	60.07 ± 0.16	86.10 ± 0.43	48.86
	Run 2	37.60 ± 1.03	25.00 ± 0.70	75.23 ± 0.89	15.73 ± 0.71	59.68 ± 0.72	86.30 ± 0.29	49.92
	Run 3	32.50 ± 0.82	25.94 ± 0.96	73.44 ± 0.85	15.94 ± 0.61	59.81 ± 0.65	86.42 ± 0.25	49.01
	Mean	**35.76** ± 2.83	24.83 ± 1.21	73.91 ± 1.16	14.97 ± 1.51	59.85 ± 0.20	86.27 ± 0.16	49.27 ± 0.57
	p vs. OPD	0.231	0.348	0.507	0.825	0.606	0.815	0.481
SCOUT	Run 1	36.35 ± 0.94	28.12 ± 0.73	76.17 ± 0.65	18.75 ± 0.55	63.25 ± 0.66	88.67 ± 0.27	51.89
	Run 2	35.52 ± 1.06	28.96 ± 0.87	75.08 ± 0.71	15.10 ± 0.54	61.57 ± 0.41	88.38 ± 0.24	50.77
	Run 3	35.42 ± 1.03	29.38 ± 0.84	76.02 ± 0.78	17.81 ± 0.71	62.31 ± 0.35	88.17 ± 0.29	51.52
	Mean	**35.76** ± 0.51	**28.82** ± 0.64	**75.76** ± 0.59	**17.22** ± 1.90	**62.38** ± 0.84	**88.41** ± 0.25	**51.39** ± 0.57
	p vs. OPD	0.192	0.0151	0.112	0.200	0.0647	0.0839	0.0598

improvement over OPD are computed as described in Appendix C.3; orange cells indicate $p < 0.1$. Best and second-best three-run aggregate results are shown in **bold** and underline, respectively.

Tables 10 and 11 correspond to the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting in Table 1. Relay-OPD fails the code generation task, as it generates excessive syntax errors. Therefore, we skipped repeated training on it. Table 12 corresponds to the Qwen3-8B-DAPO → Qwen3-1.7B setting in Table 2, and Table 13 corresponds to the Skywork-OR1-Math-7B → DeepSeek-R1-Distill-Qwen-1.5B setting in Table 3.

D.2 Results of the Analysis Experiments

We present the complete per-benchmark results of the analysis experiments. Tables 14 and 15 correspond to the ablation study on teacher GRPO conditions in Table 4. Tables 17 and 18 correspond to the analysis on whether SCOUT can complement other OPD methods in Table 5.

Table 16 corresponds to the comparison of different teacher update frequencies in Figure 4. Importantly, we find that the teacher learning rate 5×10^{-6} is too large for frequent updates, and reduce it proportionally with f, to 2.5×10^{-6} and 5×10^{-7} for f=5 and f=1, respectively. Since f=1 is time-consuming, we skip the repeat run for this experiment and report the single-run result.

Table 11: Per-run results on coding benchmarks for the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting. This table corresponds to the code generation task in Table 1. While other OPD-based methods perform worse in the code domain, SCOUT demonstrates its generalizability across domains, with significantly better accuracy on 2 of 3 benchmarks and in the average performance.

Method	**Run**	**LCB v5**	**HumanEval+**	**MBPP**	**Avg.**
	Avg.@	*4*	*16*	*8*	–
GRPO	Run 1	43.12 ± 0.65	74.16 ± 1.64	64.20 ± 0.85	60.49
	Run 2	38.81 ± 1.50	72.71 ± 2.94	70.85 ± 1.63	60.79
	Run 3	51.02 ± 1.53	71.34 ± 2.73	73.88 ± 1.57	65.41
	Mean	44.32 ± 6.20	72.74 ± 1.41	69.64 ± 4.95	62.23 ± 2.76
OPD	Run 1	37.67 ± 0.25	69.21 ± 0.82	62.50 ± 0.36	56.46
	Run 2	37.67 ± 0.55	69.51 ± 0.78	62.18 ± 0.42	56.45
	Run 3	37.59 ± 0.29	70.50 ± 0.49	62.75 ± 0.34	56.95
	Mean	37.64 ± 0.05	69.74 ± 0.68	62.48 ± 0.29	56.62 ± 0.28
ESR	Run 1	37.05 ± 0.51	69.05 ± 0.61	61.85 ± 0.31	55.98
	Run 2	33.41 ± 0.27	67.15 ± 0.87	58.80 ± 0.37	53.12
	Run 3	37.56 ± 0.36	68.06 ± 0.85	63.10 ± 0.51	56.24
	Mean	36.00 ± 2.26	68.09 ± 0.95	61.25 ± 2.21	55.11 ± 1.73
	p vs. OPD	0.832	0.961	0.781	0.865
Prune-OPD	Run 1	36.73 ± 0.22	68.94 ± 0.73	60.80 ± 0.27	55.49
	Run 2	36.70 ± 0.59	69.13 ± 0.70	61.30 ± 0.40	55.71
	Run 3	37.05 ± 0.64	70.16 ± 0.46	61.52 ± 0.58	56.24
	Mean	36.83 ± 0.19	69.41 ± 0.66	61.21 ± 0.37	55.81 ± 0.39
	p vs. OPD	0.950	0.707	0.990	0.976
Relay-OPD	Run 1	2.3 ± 0.4	14.3 ± 1.6	19.6 ± 0.9	12.07
	Run 2	–	–	–	–
	Run 3	–	–	–	–
SCOUT	Run 1	39.97 ± 0.34	73.93 ± 0.73	66.70 ± 0.61	60.20
	Run 2	43.32 ± 0.44	73.93 ± 0.36	66.77 ± 0.39	61.34
	Run 3	38.41 ± 0.40	72.41 ± 0.56	62.20 ± 0.38	57.67
	Mean	**40.57** ± 2.51	**73.42** ± 0.88	**65.22** ± 2.62	**59.74** ± 1.88
	p vs. OPD	0.0905	0.0027	0.105	0.0497

Table 12: Complete per-run results for the Qwen3-8B-DAPO → Qwen3-1.7B setting on six math reasoning benchmarks. This table corresponds to Table 2. SCOUT significantly outperforms the OPD baseline in 5 of 6 benchmarks and in average performance. These results suggest that SCOUT's gains extend to the larger-teacher setting.

Method	**Run** *Avg.@*	**AIME24** *32*	**AIME25** *32*	**AMC23** *32*	**HMMT** *32*	**Olymp.** *4*	**MATH500** *8*	**Avg6** –
GRPO	Run 1	29.06 ± 0.97	30.10 ± 0.61	73.28 ± 0.76	16.77 ± 0.69	60.80 ± 0.48	87.85 ± 0.11	49.64
	Run 2	27.71 ± 0.98	26.46 ± 0.70	67.58 ± 0.84	16.98 ± 0.48	55.46 ± 0.22	83.67 ± 0.42	46.31
	Run 3	36.04 ± 1.26	34.58 ± 0.87	79.06 ± 0.94	22.81 ± 0.54	67.00 ± 0.33	90.62 ± 0.26	55.02
	Mean	30.94 ± 4.47	30.38 ± 4.07	73.31 ± 5.74	18.85 ± 3.43	61.09 ± 5.78	87.38 ± 3.50	50.32 ± 4.39
OPD	Run 1	32.50 ± 0.85	27.08 ± 1.01	69.38 ± 0.78	14.79 ± 0.73	59.60 ± 0.64	86.52 ± 0.36	48.31
	Run 2	33.75 ± 1.08	27.29 ± 0.77	70.86 ± 0.95	15.42 ± 0.63	61.19 ± 0.86	86.98 ± 0.26	49.25
	Run 3	33.85 ± 0.78	28.96 ± 0.72	70.08 ± 0.95	17.40 ± 0.59	59.17 ± 0.25	87.55 ± 0.27	49.50
	Mean	33.37 ± 0.75	27.78 ± 1.03	70.10 ± 0.74	15.87 ± 1.36	59.98 ± 1.07	87.02 ± 0.51	49.02 ± 0.63
ESR	Run 1	35.52 ± 0.83	25.52 ± 0.87	70.86 ± 1.11	15.73 ± 0.78	60.20 ± 0.88	86.58 ± 0.29	49.07
	Run 2	36.88 ± 0.88	28.85 ± 0.99	67.19 ± 0.75	15.62 ± 0.59	60.71 ± 0.29	86.98 ± 0.24	49.37
	Run 3	32.08 ± 0.98	25.73 ± 1.01	70.31 ± 1.00	16.04 ± 0.64	61.02 ± 0.29	86.92 ± 0.16	48.68
	Mean	34.83 ± 2.47	26.70 ± 1.87	69.45 ± 1.98	15.80 ± 0.22	60.64 ± 0.41	86.83 ± 0.22	49.04 ± 0.35
	p vs. OPD	0.210	0.778	0.679	0.531	0.207	0.697	0.482
Prune-OPD	Run 1	35.00 ± 1.15	25.73 ± 0.95	69.69 ± 0.90	15.31 ± 0.72	59.55 ± 0.73	86.65 ± 0.38	48.66
	Run 2	33.44 ± 1.14	25.83 ± 1.02	69.69 ± 0.98	18.02 ± 0.70	58.30 ± 0.40	85.90 ± 0.26	48.53
	Run 3	32.40 ± 1.07	26.88 ± 0.81	69.30 ± 0.94	16.25 ± 0.70	58.82 ± 0.38	86.52 ± 0.29	48.36
	Mean	33.61 ± 1.31	26.15 ± 0.64	69.56 ± 0.23	16.53 ± 1.38	58.89 ± 0.63	86.36 ± 0.40	48.52 ± 0.15
	p vs. OPD	0.402	0.944	0.747	0.294	0.892	0.920	0.852
Relay-OPD	Run 1	31.46 ± 1.01	27.40 ± 0.91	71.17 ± 0.84	16.46 ± 0.58	60.07 ± 0.25	87.25 ± 0.36	48.97
	Run 2	31.04 ± 0.71	26.04 ± 1.18	71.02 ± 0.78	15.31 ± 0.70	59.68 ± 0.48	86.92 ± 0.24	48.34
	Run 3	31.15 ± 1.04	28.44 ± 0.91	69.14 ± 0.99	14.27 ± 0.67	58.69 ± 0.21	87.47 ± 0.38	48.19
	Mean	31.22 ± 0.22	27.29 ± 1.20	70.44 ± 1.13	15.35 ± 1.09	59.48 ± 0.71	87.21 ± 0.28	48.50 ± 0.41
	p vs. OPD	0.977	0.687	0.354	0.684	0.732	0.307	0.848
SCOUT	Run 1	31.35 ± 1.15	30.31 ± 0.93	74.45 ± 0.85	16.56 ± 0.69	61.57 ± 0.64	88.65 ± 0.30	50.48
	Run 2	39.17 ± 0.99	31.56 ± 0.92	75.70 ± 0.80	18.85 ± 0.73	61.02 ± 0.11	88.55 ± 0.43	52.48
	Run 3	35.62 ± 0.72	31.56 ± 0.70	74.84 ± 0.97	17.71 ± 0.53	61.45 ± 0.54	89.72 ± 0.27	51.82
	Mean	**35.38** ± 3.92	**31.15** ± 0.72	**75.00** ± 0.64	**17.71** ± 1.15	**61.35** ± 0.29	**88.97** ± 0.65	**51.59** ± 1.02
	p vs. OPD	0.235	0.00652	0.00125	0.0752	0.0720	0.00828	0.0140

Table 13: Complete per-run results for the Skywork-OR1-Math-7B → DeepSeek-R1-Distill-Qwen-1.5B setting on six math reasoning benchmarks. This table corresponds to Table 3. SCOUT significantly outperforms OPD on 4 of 6 benchmarks and in average performance, which extends its generalizability beyond the Qwen model family.

Method	**Run** *Avg.@*	**AIME24** *32*	**AIME25** *32*	**AMC23** *32*	**HMMT** *32*	**Olymp.** *4*	**MATH500** *8*	**Avg6** –
OPD	Run 1	37.92 ± 1.05	30.52 ± 0.73	77.27 ± 0.77	19.17 ± 0.89	60.46 ± 0.76	89.00 ± 0.26	52.39
	Run 2	38.33 ± 0.89	29.90 ± 0.68	78.44 ± 0.66	19.06 ± 0.45	60.80 ± 0.76	89.15 ± 0.25	52.61
	Run 3	37.08 ± 1.06	29.58 ± 0.53	78.05 ± 0.94	18.96 ± 0.55	60.41 ± 0.64	89.10 ± 0.23	52.20
	Mean	37.78 ± 0.64	**30.00** ± 0.48	77.92 ± 0.60	**19.06** ± 0.11	60.56 ± 0.21	89.08 ± 0.08	52.40 ± 0.21
ESR	Run 1	31.67 ± 1.20	27.40 ± 0.83	73.44 ± 0.82	18.44 ± 0.67	56.80 ± 0.61	87.38 ± 0.36	49.19
	Run 2	31.15 ± 1.02	27.19 ± 0.75	70.23 ± 0.96	16.98 ± 0.53	57.27 ± 0.69	87.62 ± 0.32	48.41
	Run 3	34.90 ± 1.15	27.40 ± 0.79	71.09 ± 0.88	16.35 ± 0.59	56.50 ± 0.51	87.38 ± 0.40	48.94
	Mean	32.57 ± 2.03	27.33 ± 0.12	71.59 ± 1.66	17.26 ± 1.07	56.86 ± 0.39	87.46 ± 0.14	48.84 ± 0.40
	p vs. Frozen-OPD	0.985	0.994	0.995	0.960	0.999	0.998	1.000
Prune-OPD	Run 1	22.60 ± 1.01	19.17 ± 0.73	56.88 ± 0.83	10.94 ± 0.62	39.93 ± 0.73	72.12 ± 0.36	36.94
	Run 2	20.52 ± 1.05	18.44 ± 0.81	56.48 ± 0.83	11.46 ± 0.70	41.22 ± 0.29	72.28 ± 0.50	36.73
	Run 3	21.67 ± 0.70	18.85 ± 0.84	56.95 ± 1.03	11.04 ± 0.71	40.32 ± 0.46	71.47 ± 0.60	36.72
	Mean	21.60 ± 1.04	18.82 ± 0.37	56.77 ± 0.25	11.15 ± 0.28	40.49 ± 0.66	71.96 ± 0.43	36.80 ± 0.12
	p vs. Frozen-OPD	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Relay-OPD	Run 1	32.81 ± 1.22	28.65 ± 0.80	74.84 ± 0.87	16.04 ± 0.61	58.30 ± 0.49	88.47 ± 0.28	49.85
	Run 2	35.10 ± 0.91	27.19 ± 0.79	72.97 ± 0.69	17.29 ± 0.55	58.35 ± 0.81	88.35 ± 0.24	49.88
	Run 3	36.25 ± 1.11	29.17 ± 0.70	76.09 ± 0.97	15.73 ± 0.64	60.03 ± 0.44	88.08 ± 0.20	50.89
	Mean	34.72 ± 1.75	28.34 ± 1.03	74.63 ± 1.57	16.35 ± 0.83	58.89 ± 0.98	88.30 ± 0.20	50.21 ± 0.59
	p vs. Frozen-OPD	0.963	0.956	0.976	0.994	0.958	0.991	0.992
SCOUT	Run 1	40.42 ± 1.08	29.90 ± 0.71	82.03 ± 0.84	18.33 ± 0.58	61.62 ± 0.86	90.08 ± 0.36	53.73
	Run 2	39.90 ± 0.90	30.52 ± 0.78	80.23 ± 0.67	18.96 ± 0.68	62.13 ± 0.61	90.05 ± 0.26	53.63
	Run 3	39.06 ± 1.38	29.58 ± 0.61	81.17 ± 0.82	18.75 ± 0.74	61.75 ± 0.55	90.00 ± 0.22	53.39
	Mean	**39.79** ± 0.69	**30.00** ± 0.48	**81.14** ± 0.90	18.68 ± 0.32	**61.83** ± 0.26	**90.04** ± 0.04	**53.58** ± 0.18
	p vs. Frozen-OPD	0.0418	0.500	0.00504	0.741	0.0454	0.00613	< 0.001

Table 14: Per-run mathematical-reasoning results for the **OPD + Teacher GRPO** control. This control applies the same outcome-reward GRPO update to the teacher as SCOUT, but the teacher rolls out from scratch without conditioning on a student-generated prefix. OPD and SCOUT are shown as three-run aggregates for reference. For individual training runs, the smaller ± term denotes within-evaluation standard error; in *Mean* rows, it denotes standard deviation across three training runs. Avg. is the macro-average over the six displayed math benchmarks. Best and second-best three-run aggregates are shown in **bold** and underline, respectively.

Method	**Run** *Avg.@*	**AIME24** *32*	**AIME25** *32*	**AMC23** *32*	**HMMT** *32*	**Olymp.** *4*	**MATH500** *8*	**Avg.** –
(a) Qwen3-4B-Instruct-2507 → Qwen3-1.7B								
OPD	Mean	33.82 ± 3.03	24.31 ± 1.75	73.93 ± 1.85	16.04 ± 0.75	60.15 ± 1.64	87.03 ± 1.14	49.21 ± 1.53
OPD + Teacher GRPO	Run 1	29.17 ± 0.82	23.65 ± 0.68	73.05 ± 0.87	13.96 ± 0.55	58.56 ± 0.55	86.67 ± 0.30	47.51
	Run 2	32.29 ± 1.05	26.56 ± 0.73	74.53 ± 0.96	16.04 ± 0.66	60.15 ± 0.72	87.83 ± 0.17	49.57
	Run 3	32.50 ± 1.07	27.40 ± 0.83	71.95 ± 0.92	17.71 ± 0.72	59.42 ± 0.46	87.12 ± 0.33	49.35
	Mean	31.32 ± 1.86	25.87 ± 1.97	73.18 ± 1.29	15.90 ± 1.88	59.38 ± 0.80	87.21 ± 0.58	48.81 ± 1.13
SCOUT	Mean	**35.76** ± 0.51	**28.82** ± 0.64	**75.76** ± 0.59	**17.22** ± 1.90	**62.38** ± 0.84	**88.41** ± 0.25	**51.39** ± 0.57
(b) Qwen3-8B-DAPO → Qwen3-1.7B								
OPD	Mean	33.37 ± 0.75	27.78 ± 1.03	70.10 ± 0.74	15.87 ± 1.36	59.98 ± 1.07	87.02 ± 0.51	49.02 ± 0.61
OPD + Teacher GRPO	Run 1	38.54 ± 1.01	32.40 ± 0.81	74.45 ± 0.78	17.60 ± 0.64	61.10 ± 0.41	88.90 ± 0.23	52.17
	Run 2	34.17 ± 0.99	29.90 ± 0.71	72.42 ± 0.81	16.77 ± 0.69	60.97 ± 0.66	87.52 ± 0.24	50.29
	Run 3	32.29 ± 0.87	31.77 ± 0.81	71.56 ± 0.79	16.35 ± 0.69	59.85 ± 0.44	87.67 ± 0.40	49.92
	Mean	35.00 ± 3.21	**31.36** ± 1.30	72.81 ± 1.48	16.91 ± 0.64	60.64 ± 0.69	88.03 ± 0.76	50.79 ± 1.21
SCOUT	Mean	**35.38** ± 3.91	31.15 ± 0.72	**75.00** ± 0.64	**17.71** ± 1.15	**61.35** ± 0.29	**88.97** ± 0.65	**51.59** ± 1.02

Table 15: Per-run code-generation results for the **OPD + Teacher GRPO** control in the Qwen3-4B-Instruct-2507 → Qwen3-1.7B setting. This control applies the same outcome-reward GRPO update to the teacher as SCOUT, but the teacher rolls out from scratch without conditioning on a student-generated prefix. OPD and SCOUT are shown as three-run aggregates for reference. For individual training runs, the smaller ± term denotes within-evaluation standard error; in *Mean* rows, it denotes standard deviation across three training runs. Best and second-best three-run aggregates are shown in **bold** and underline, respectively.

Method	**Run** *Avg.@*	**LCB v5** *4*	**HumanEval+** *16*	**MBPP** *8*	**Avg.** –
OPD	Mean	37.64 ± 0.05	69.74 ± 0.68	62.48 ± 0.29	56.62 ± 0.28
OPD + Teacher GRPO	Run 1	38.01 ± 0.30	69.21 ± 0.50	61.35 ± 0.49	56.19
	Run 2	39.40 ± 0.29	72.18 ± 0.49	65.80 ± 0.27	59.13
	Run 3	36.08 ± 0.24	69.36 ± 0.60	61.72 ± 0.63	55.72
	Mean	37.83 ± 1.67	70.25 ± 1.67	62.96 ± 2.47	57.01 ± 1.85
SCOUT	Mean	**40.57** ± 2.51	**73.42** ± 0.88	**65.22** ± 2.62	**59.74** ± 1.88

Table 16: Ablation on the teacher update interval f with the Qwen3-8B-DAPO → Qwen3-1.7B student setting. Updating the teacher with $f \leq 10$ consistently improves over OPD, whereas the improvement becomes marginal or disappears for $f \geq 15$. Although $f = 5$ achieves the highest average accuracy, $f = 10$ retains most of the improvement with fewer teacher updates, providing a favorable performance–computation trade-off.

Model	AIME24	AIME25	AMC23	HMMT	Olympiad	MATH-500	Mean
Qwen3-1.7B	13.3±4.2	10.3±3.2	46.8±4.9	5.4±2.6	42.3±1.5	73.1±1.5	31.9
Qwen3-8B-DAPO	59.9±6.0	43.9±5.9	90.7±3.9	25.7±4.1	74.2±0.7	94.6±0.5	64.8
OPD	33.5±6.5	28.0±5.5	69.2±4.9	16.4±4.1	59.9±0.4	87.2±0.9	49.0
SCOUT (f=1)	33.0±5.8	29.8±4.6	**76.1**±4.8	18.2±3.9	**62.9**±1.0	87.8±0.8	51.3
SCOUT (f=5)	38.4±6.0	**30.3**±5.4	75.8±5.2	**18.3**±4.1	62.7±1.8	**89.6**±0.6	**52.5**
SCOUT (f=10)	**39.2**±5.0	28.7±6.1	74.2±5.0	16.5±4.1	60.8±0.4	89.4±0.7	51.5
SCOUT (f=15)	34.3±4.9	27.7±5.0	70.7±5.8	15.3±4.2	60.0±1.0	87.2±1.1	49.2
SCOUT (f=20)	33.5±6.2	29.2±4.7	68.0±5.1	15.4±3.5	59.7±1.3	86.7±1.1	48.8

Table 17: Full results for combining SCOUT with On-policy trust region OPD (OPTR) in the Qwen3-4B-Instruct-2507 → Qwen3-1.7B math setting. OPTR performs better than OPD, and `SCOUT` further improves OPTR when combined with it. This shows that `SCOUT` is complementary to other loss-level OPD methods.

Method	**Run** *Avg.@*	**AIME24** *32*	**AIME25** *32*	**AMC23** *32*	**HMMT** *32*	**Olymp.** *4*	**MATH500** *8*	**Avg.** –
OPD	Mean	33.82 ± 3.03	24.31 ± 1.75	73.93 ± 1.85	16.04 ± 0.75	60.15 ± 1.64	87.03 ± 1.14	49.21 ± 1.53
SCOUT	Mean	35.76 ± 0.51	**28.82** ± 0.64	75.76 ± 0.59	**17.22** ± 1.90	62.38 ± 0.84	88.41 ± 0.25	51.39 ± 0.57
OPTR	Run 1	33.02 ± 1.01	24.06 ± 0.86	76.56 ± 0.82	17.50 ± 0.75	61.14 ± 0.56	86.92 ± 0.44	49.87
	Run 2	32.81 ± 0.95	24.58 ± 0.74	73.98 ± 0.85	15.52 ± 0.66	61.49 ± 0.43	86.80 ± 0.49	49.20
	Run 3	34.90 ± 0.92	26.25 ± 0.79	75.08 ± 0.66	16.67 ± 0.75	61.66 ± 0.37	87.35 ± 0.28	50.32
	Mean	33.58 ± 1.15	24.96 ± 1.14	75.21 ± 1.29	16.56 ± 0.99	61.43 ± 0.27	87.02 ± 0.29	49.80 ± 0.56
OPTR + SCOUT	Run 1	37.60 ± 0.98	27.50 ± 0.86	77.89 ± 0.78	15.83 ± 0.67	63.51 ± 0.98	89.70 ± 0.40	52.01
	Run 2	37.08 ± 1.12	27.19 ± 0.64	80.00 ± 0.89	18.65 ± 0.67	63.64 ± 0.57	89.48 ± 0.26	52.67
	Run 3	34.79 ± 0.73	28.23 ± 0.86	78.83 ± 0.77	17.08 ± 0.70	63.77 ± 0.22	90.10 ± 0.22	52.13
	Mean	**36.49** ± 1.50	27.64 ± 0.53	**78.91** ± 1.06	17.19 ± 1.41	**63.64** ± 0.13	**89.76** ± 0.31	**52.27** ± 0.35

Table 18: Full results for combining SCOUT with On-policy trust region OPD (OPTR) in the Qwen3-4B-Instruct-2507 $\rightarrow$ Qwen3-1.7B code setting. When combined with SCOUT, OPTR achieves further improvement, showing the same trend as Table 17. This proves that the complementary merit of SCOUT generalizes to the code domain.

Method	**Run** *Avg.@*	**LCB v5** *4*	**HumanEval+** *16*	**MBPP** *8*	**Avg.** –
OPD	Mean	37.64 ± 0.05	69.74 ± 0.68	62.48 ± 0.29	56.62 ± 0.28
SCOUT	Mean	40.57 ± 2.51	<u>73.42</u> ± 0.88	<u>65.22</u> ± 2.62	<u>59.74</u> ± 1.88
OPTR	Run 1	41.42 ± 0.64	73.67 ± 0.52	63.77 ± 0.36	59.62
	Run 2	40.40 ± 0.26	72.56 ± 0.94	64.58 ± 0.56	59.18
	Run 3	41.42 ± 0.38	72.79 ± 0.72	63.45 ± 0.71	59.22
	Mean	<u>41.08</u> ± 0.59	73.01 ± 0.58	63.93 ± 0.58	59.34 ± 0.24
OPTR + SCOUT	Run 1	42.95 ± 0.46	75.08 ± 0.49	66.38 ± 0.50	61.47
	Run 2	40.99 ± 0.45	73.48 ± 0.64	63.58 ± 0.52	59.35
	Run 3	41.53 ± 0.55	74.31 ± 0.74	66.47 ± 0.18	60.77
	Mean	**41.83** ± 1.01	**74.29** ± 0.80	**65.47** ± 1.65	**60.53** ± 1.08