REOPENING REINFORCEMENT LEARNING HEADROOM IN LANGUAGE MODELS WITH ITERATIVE RESET ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards can substantially improve language-model reasoning, yet continued optimization often enters a low-gain regime. We introduce **Iterative Reset On-Policy Distillation and Reinforcement Learning** (**IR-OPD-RL**), which repeatedly resets a student to a fixed supervised fine-tuning checkpoint, transfers the current teacher's behavior through reward-extrapolated G-OPD, and launches a new GRPO stage from the resulting student. This design transfers behavioral knowledge without inheriting the teacher's parameters or optimizer history. Across Qwen3-1.7B and Qwen3-4B on six mathematical-reasoning and code-generation configurations, IR-OPD-RL improves domain-average accuracy over the initial RL teachers by 3.30–8.23 points. On Qwen3-4B, IR-OPD-RL further exceeds ExOPD, the strongest tested single-stage baseline, by 5.42 points on Math-57K and 2.45 points on Code-25K. Under matched compute on Qwen3-1.7B, IR-OPD-RL outperforms continued RL by up to 2.35 points on Math-57K and 3.70 points on Code-25K. Order and initialization ablations further favor distillation before RL and resetting the student from the fixed SFT anchor. These results show that immediate checkpoint accuracy need not predict subsequent RL improvement and that reset on-policy distillation can renew optimization headroom.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a central approach to post-training language models for mathematical reasoning and code generation. By directly optimizing answer correctness or execution outcomes, methods such as Group Relative Policy Optimization (GRPO) improve task performance without dense human annotations (Shao et al., 2024; DeepSeek-AI, 2025). Continued RL, however, is not uniformly productive: as training progresses, additional rollout and optimization compute may yield little improvement despite substantial remaining task error. This motivates considering not only a checkpoint's current accuracy but also its **downstream RL improvability**, defined as the endpoint reached under a fixed subsequent RL procedure and compute budget. A strong checkpoint may therefore be a poor initialization for further reward optimization, whereas a weaker checkpoint may ultimately reach a stronger endpoint.

On-policy distillation (OPD) provides a complementary mechanism for transferring policy behavior. The student generates its own trajectories, while the teacher provides dense token-level supervision at student-visited prefixes (Agarwal et al., 2024). Iterated Relearning (ITER) in deep RL follows a related reset-and-relearn paradigm: a trained policy is repeatedly transferred to a freshly initialized student before RL is resumed, reducing the harmful effects of transient training non-stationarity and improving generalization (Igl et al., 2021). However, ITER studies visual-control RL, while existing language-model distillation–RL methods mainly use joint optimization or a one-time handoff (Xu et al., 2025; Ye et al., 2026; Chen et al., 2025; Rang et al., 2026). It remains unclear whether repeatedly transferring an RL reasoning policy into a student reset from a shared SFT anchor can improve subsequent reward optimization.

We address this question with **Iterative Reset On-Policy Distillation and Reinforcement Learning** (**IR-OPD-RL**). At each round, a student is reinitialized from a fixed supervised fine-tuning

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.

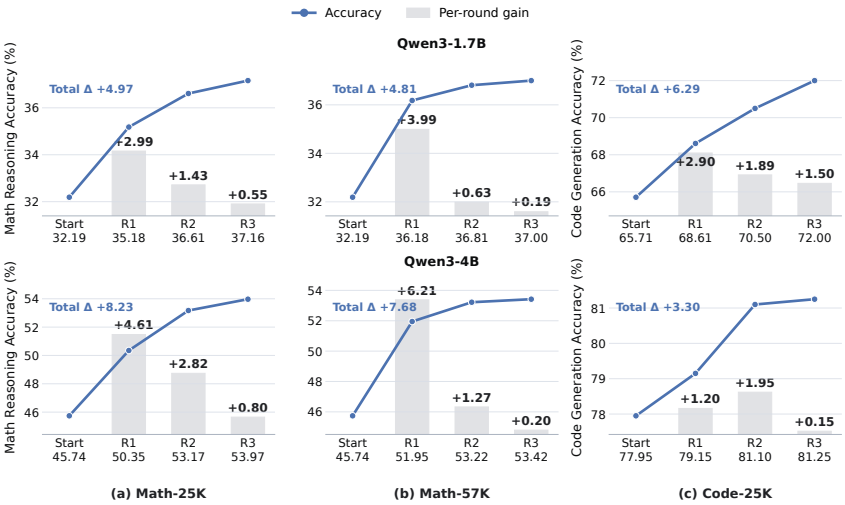


Figure 1: **Accuracy trajectories and per-round gains of IR-OPD-RL.** Results are shown for Qwen3-1.7B and Qwen3-4B on Math-25K, Math-57K, and Code-25K. Blue curves track absolute accuracy from the initial RL teacher (Start) through three G-OPD+RL rounds, while translucent bars report the net accuracy gain from each complete round. Accuracy axes are scaled independently in each panel.

(SFT) checkpoint and learns the current teacher's behavior through reward-extrapolated G-OPD. The resulting student initializes a new GRPO stage, whose output becomes the teacher for the next round. IR-OPD-RL thus propagates information across rounds through behavior while avoiding direct inheritance of teacher parameters and optimizer state.

We evaluate IR-OPD-RL with Qwen3-1.7B and Qwen3-4B on Math-25K, Math-57K, and Code-25K. As summarized in Figure 1, after three rounds it improves domain-average accuracy over the initial RL teachers by **3.30**–**8.23** points across all six model–data configurations. Under matched compute on Qwen3-1.7B, it outperforms continued RL by **1.84**–**2.35** points across three Math-57K budgets. On Code-25K, the methods are comparable after the first round, after which IR-OPD-RL gains increase to **1.65** and **3.70** points. Additional controls favor G-OPD before RL, repeated alternation over a one-shot handoff, and resetting from the fixed SFT anchor.

Our contributions are:

1. We introduce **IR-OPD-RL**, a multi-round post-training framework that alternates reset G-OPD with renewed RL, separating behavioral transfer from parameter and optimizer-state inheritance.
2. Across two model scales, two verifiable domains, and six model–data configurations, IR-OPD-RL improves strong initial RL teachers by **3.30**–**8.23** accuracy points. On Qwen3-4B, it also exceeds the strongest tested single-stage baseline, ExOPD, by **5.42** points on Math-57K and **2.45** points on Code-25K.
3. Matched-compute comparisons and controlled ablations show that the gains cannot be explained solely by additional RL compute or optimizer restart. IR-OPD-RL consistently outperforms continued RL on Math-57K and develops increasing advantages on Code-25K, while order, alternation, and initialization studies favor applying G-OPD before RL and resetting from the fixed SFT anchor.

2 Related Work

2.1 Reinforcement Learning for Verifiable Reasoning

Reinforcement learning with verifiable rewards (RLVR) optimizes automatically checked outcomes, with GRPO-based systems demonstrating strong reasoning gains without dense human labels (Shao et al., 2024; DeepSeek-AI, 2025). PPO underlies many policy-gradient pipelines (Schulman et al.,

2017), while recent rule-based systems expose the importance of sampling, clipping, reward design, and GRPO bias (Xie et al., 2025; Yu et al., 2025; Liu et al., 2025b). Because outcome rewards are sparse, process supervision offers denser step-level feedback (Setlur et al., 2024; Lightman et al., 2023); the interpretation of RLVR's capability gains also remains debated (Yue et al., 2025; Wen et al., 2025). We therefore focus on whether a training schedule can improve the next RL stage, rather than on resolving that broader capability question.

2.2 On-Policy Distillation and RL

Knowledge and policy distillation transfer teacher behavior to a student (Hinton et al., 2015; Rusu et al., 2016); OPD instead supervises student-generated trajectories, which helps address the distribution shift emphasized in iterative imitation learning (Ross et al., 2011) and supplies token-level supervision on the student's visited states (Xiao et al., 2025; Agarwal et al., 2024). G-OPD formulates this transfer as dense KL-regularized policy optimization with reward scaling (Yang et al., 2026). Related approaches combine distillation and RL through joint objectives, one-shot handoffs, offline re-distillation, or parameter reset (Xu et al., 2025; Ye et al., 2026; Chen et al., 2025; Liu et al., 2026); IR-OPD-RL instead repeatedly resets the student to a fixed SFT anchor and refreshes the teacher after every RL stage.

2.3 Multi-Round Teacher–Student Evolution

Iterative policy-improvement loops also appear in Expert Iteration, which alternates search with policy learning (Anthony et al., 2017), and in STaR, which bootstraps reasoning from self-generated rationales (Zelikman et al., 2022). These methods use different supervision and do not study repeated reset G-OPD-RL handoffs.

MOPD extends on-policy distillation to multiple rounds by integrating domain-specific RL experts into a shared student and using that student to initialize the next round of expert training (Ma et al., 2026). Its objective is multi-domain capability integration, and model parameters are inherited across rounds. IR-OPD-RL instead studies a single-domain reset setting: every student begins from the same fixed SFT anchor, while only the behavior of the latest same-task RL policy is transferred. This isolates repeated behavioral consolidation from direct parameter and optimizer-state inheritance.

3 Method

We introduce **Iterative Reset On-Policy Distillation and Reinforcement Learning** (**IR-OPD-RL**), which organizes post-training as an alternating process of policy consolidation and reward-driven improvement. As illustrated in Figure 2, each round reinitializes a student from the same supervised fine-tuning (SFT) anchor, transfers the behavior of the latest RL policy to this student through G-OPD, and then starts a new RL stage from the resulting student. The improved policy becomes the teacher for the next round, forming a recurrent reset–G-OPD–improvement cycle.

3.1 Method Overview

We consider mathematical reasoning and code-generation tasks with verifiable rewards. Let $x \sim \mathcal{D}_{\text{task}}$ denote a prompt sampled from a task-specific training set, $y \sim \pi_\theta(\cdot \mid x)$ a response generated by policy π_θ. Let $r(x, y) \in \{0, 1\}$ denote the verifiable reward determined by answer correctness or code execution. For each model scale m, we fix an SFT checkpoint B_m as the parameter anchor shared by all rounds.

Let T_k denote the RL policy available at the beginning of round k, and let S_{k+1} denote the student produced in that round. Each round consists of two stages:

$$S_{k+1} = \text{G-OPD}_\lambda \left(B_m, T_k; \mathcal{D}_{\text{task}}\right), \tag{1}$$

$$T_{k+1} = \text{GRPO}\left(S_{k+1}; \mathcal{D}_k^{\text{RL}}\right). \tag{2}$$

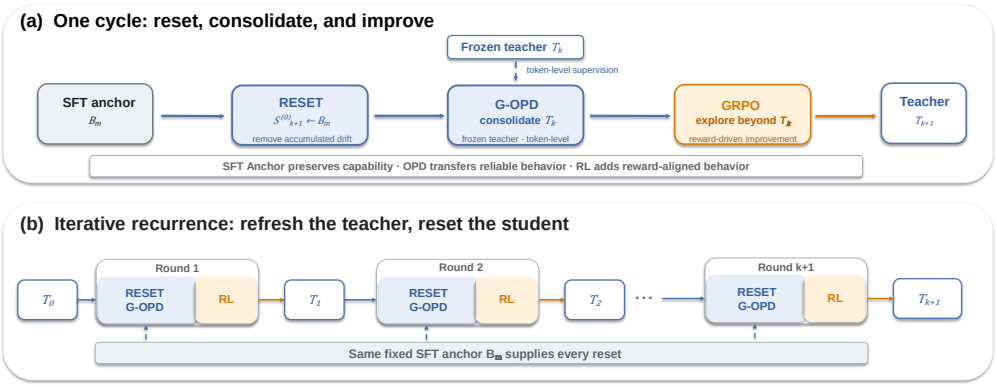


Figure 2: **Overview of IR-OPD-RL.** (a) At round k, a fresh student and optimizer are reset from the fixed SFT anchor B_m, while the frozen policy T_k transfers behavior only through token-level supervision. G-OPD produces the student S_{k+1}, which initializes GRPO and yields the improved policy T_{k+1}. (b) Every round resets the student from the same SFT anchor while refreshing the teacher with the policy produced by the preceding round. Behavioral knowledge is therefore accumulated iteratively without inheriting teacher parameters or optimizer history.

In the first stage, the student parameters are initialized from B_m, and a fresh optimizer is created. The current teacher T_k remains frozen and provides dense token-level supervision on trajectories generated by the student. G-OPD is performed over the full task dataset $\mathcal{D}_{\text{task}}$, consolidating the teacher's behavior into a student with a different parameter state and optimization history. In the second stage, S_{k+1} initializes a new GRPO run. Its output T_{k+1} is both the result of the current round and the teacher used in the next round.

The initial teacher T_0 is obtained through a preparatory domain post-training stage. The formal recurrence begins from T_0 and refreshes the teacher with the latest RL output after every round. The exact teacher configuration is an experimental choice and does not change the reset and behavior-transfer mechanism in Eq. 1. Across all rounds, the student parameters return to the fixed anchor B_m; only teacher behavior is transferred.

3.2 Reset G-OPD and Policy Improvement

Reset G-OPD. At the beginning of every round, the student is initialized from B_m rather than from the parameters of T_k, and the optimizer is initialized from scratch. This reset is therefore different from an optimizer restart, which retains the current policy parameters. The student generates responses under its current policy, while the frozen teacher and SFT anchor evaluate the same tokens at prefixes visited by the student. The supervision is consequently on-policy and does not depend on offline teacher-generated trajectories.

We use Generalized On-Policy Distillation (G-OPD) as the policy-consolidation operator and set the reward-scaling coefficient to $\lambda = 1.25$ (Yang et al., 2026), corresponding to its reward-extrapolated configuration. For a student-generated token a_t at state s_t, the token-level training signal is

$$A_t^{\text{G-OPD}} = \lambda \log \pi_{T_k}(a_t \mid s_t) + (1-\lambda) \log \pi_{B_m}(a_t \mid s_t) - \log \pi_{\theta^-}(a_t \mid s_t), \tag{3}$$

where π_{θ^-} is the student policy that generated the current trajectory. When $\lambda > 1$, this signal extrapolates the teacher's policy shift relative to the SFT anchor. G-OPD is intended to construct a new initialization for subsequent reward optimization; the immediate accuracy of the resulting student is therefore not required to exceed that of its teacher.

RL policy improvement. After G-OPD, we directly initialize GRPO (Shao et al., 2024) from S_{k+1}. GRPO samples a group of responses for each prompt and estimates advantages from their relative outcome rewards. We use answer correctness for mathematical reasoning and unit-test outcomes for code generation. Crucially, this stage optimizes from the parameter state produced by

Algorithm 1 Iterative Reset G-OPD-RL

Require: Fixed SFT anchor B_m; initial teacher T_0; shared task dataset $\mathcal{D}_{\text{task}}$; number of rounds K; optional data-selection operator $\mathcal{F}_k$

for $k = 0, \ldots, K-1$ **do**
  Initialize student S_{k+1} from B_m with a fresh optimizer
  Freeze the current teacher T_k
  Train S_{k+1} with G-OPD on $\mathcal{D}_{\text{task}}$
  if data selection is enabled **then**
    $\mathcal{D}_k^{\text{RL}} \leftarrow \mathcal{F}_k(\mathcal{D}_{\text{task}}, S_{k+1})$
  else
    $\mathcal{D}_k^{\text{RL}} \leftarrow \mathcal{D}_{\text{task}}$
  end if
  Initialize GRPO from S_{k+1}
  Train with GRPO on $\mathcal{D}_k^{\text{RL}}$ to obtain T_{k+1}
end for
return T_K

G-OPD instead of continuing from T_k. Its output is denoted by T_{k+1} and is frozen as the teacher in the next round.

G-OPD and GRPO thus play complementary roles. G-OPD consolidates the current teacher's behavior through dense token-level supervision, whereas GRPO further improves the policy using verifiable sequence-level rewards. Repeating these stages tests whether transferring the behavior of a saturated RL policy into a reset student can reopen optimization headroom.

3.3 Iterative Training Procedure

Algorithm 1 summarizes the complete training procedure. G-OPD uses the full task dataset $\mathcal{D}_{\text{task}}$ in every round, whereas GRPO uses $\mathcal{D}_k^{\text{RL}}$, which is either the same dataset or an optionally selected subset of it. When enabled, checkpoint- specific selection concentrates RL updates on prompts with informative reward variation, reducing rollout and training compute and potentially reaching a target performance with fewer resources. This selection is an optional efficiency intervention rather than a required component of IR-OPD-RL; its criterion and measured savings are reported in Section 5.5. We use prespecified G-OPD and GRPO update budgets rather than adaptive stopping based on an online plateau detector. Additional optimization and rollout-correction details are provided in the experimental setup.

4 Experimental Setup

Base Models and Domain Teachers. We use Qwen3-1.7B-Non-Thinking and Qwen3-4B-Non-Thinking (Yang et al., 2025). For each model scale, the fixed SFT anchor B_m initializes every reset student and serves as the reference policy for G-OPD. We construct separate math and code teachers by applying GRPO to the corresponding domain data. Teachers and students within each model scale share the same architecture and originate from the same anchor checkpoint.

Training Data. We construct **Math-57K** by retaining DeepMath examples (He et al., 2025) with difficulty levels of at least 6. **Math-25K** is a fixed random subset of 25K examples from Math-57K. For code generation, we use the 25K examples from Eurus-RL-Code (Cui et al., 2025), denoted as **Code-25K**. G-OPD uses the full corresponding domain dataset. GRPO uses the same source data unless a checkpoint-specific filtered subset is explicitly stated.

Training. We train domain teachers with GRPO (Shao et al., 2024), using binary rewards for answer correctness in math and unit-test success in code. IR-OPD-RL runs for three rounds, each resetting the G-OPD student to B_m with a fresh optimizer and initializing GRPO from the resulting checkpoint. G-OPD uses $\lambda = 1.25$, while GRPO samples eight rollouts per prompt. Both stages use token-level rollout correction (Liu et al., 2025a) and are implemented with `verl` (Sheng et al., 2025). Full optimization hyperparameters are provided in the appendix. Compute is measured in 8-NPU node-hours, where one unit is one wall-clock hour on eight Ascend 910B NPUs.

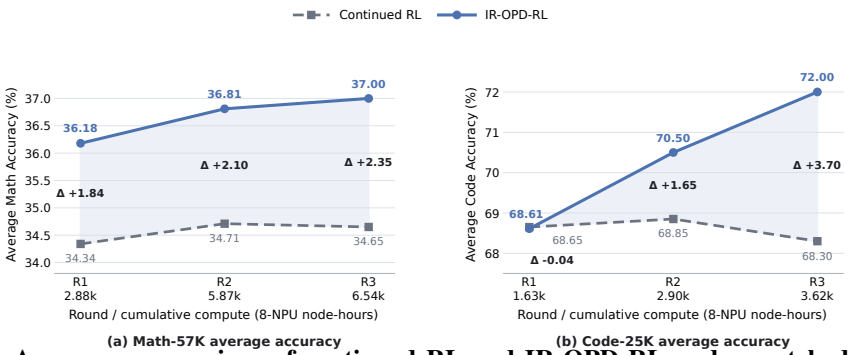


Figure 3: **Accuracy comparison of continued RL and IR-OPD-RL under matched compute across two domains on Qwen3-1.7B.** Left: average accuracy over AIME24/25 and HMMT25 (Feb./Nov.) for Math-57K. Right: average accuracy over HumanEval+ and MBPP+ for Code-25K. The two training strategies are evaluated at three matched cumulative-compute budgets in each domain. Compute is reported in 8-NPU node-hours.

Evaluation. We evaluate mathematical reasoning on four competition-level benchmarks: AIME24 (AI-MO, 2024), AIME25 (OpenCompass, 2025), HMMT25 February (Balunović et al., 2026), and HMMT25 November (Balunović et al., 2026). For code generation, we evaluate on HumanEval+ and MBPP+ (Liu et al., 2023). We use a temperature of 1.0, top-p of 1.0, and a maximum generation length of 16,384 tokens. For each problem, we sample 64 independent responses, average their binary correctness, and then average over problems; the reported metric is therefore mean sample accuracy rather than pass@64. Math responses are checked with Math-Verify, and code responses with the official EvalPlus tests.

5 Experimental Results

5.1 Performance across Model Scales, Domains, and Training Rounds

We first examine whether IR-OPD-RL consistently improves strong RL policies across model scales and task domains. As shown in Table 1, the initial RL teachers already substantially outperform their SFT anchors. After three G-OPD–RL rounds, the final policy achieves the best domain-average accuracy in all six model–data configurations. Qwen3-1.7B gains $+4.97$, $+4.81$, and $+6.29$ points on Math-25K, Math-57K, and Code-25K, respectively; the corresponding gains for Qwen3-4B are $+8.23$, $+7.68$, and $+3.30$ points. IR-OPD-RL therefore scales across both model sizes and both task domains.

Figure 1 shows that the gains are positive through all three rounds but vary by task. Math improvements are front-loaded: the first round contributes between $+2.99$ and $+6.21$ points across the four math configurations, while every third-round gain is at most $+0.80$ points. Code exhibits a less uniform pattern. Qwen3-1.7B retains substantial gains across all three rounds ($+2.90$, $+1.89$, and $+1.50$), whereas Qwen3-4B obtains its largest gain in the second round ($+1.20$, $+1.95$, and $+0.15$). Thus, additional rounds remain useful, although their marginal benefit generally diminishes.

5.2 Comparison with Existing Post-Training Methods

We compare IR-OPD-RL with three representative post-training methods on Qwen3-4B-Non-Thinking: model extrapolation (ExPO), standard on-policy distillation (OPD), and reward-extrapolated OPD (ExOPD). The RL teacher and SFT student are included as reference checkpoints. As shown in Table 2, IR-OPD-RL achieves the best result on every benchmark, exceeding the strongest baseline, ExOPD, by 5.42 points on Math-57K and 2.45 points on Code-25K.

These results demonstrate an endpoint-performance advantage over the tested single-stage methods. Because IR-OPD-RL uses three training rounds, this comparison does not establish a compute-efficiency advantage; matched-compute results are reported in Section 5.3.

Table 1: **Stage-wise performance of IR-OPD-RL.** All values are accuracies (%). Math Avg. averages AIME24/25 and HMMT25 (Feb./Nov.); Code Avg. averages HumanEval+ and MBPP+. For Math-25K, round-1 RL uses Pass-1–7, while G-OPD uses Full-25K. Bold indicates the best result within each model block.

Training		Math-25K					Math-57K					Code-25K		
Round	Stage	A24	A25	H25-Feb.	H25-Nov.	Avg.	A24	A25	H25-Feb.	H25-Nov.	Avg.	HE+	MBPP+	Avg.
Qwen3-1.7B-Non-Thinking														
SFT Anchor		12.92	11.74	5.14	3.77	8.39	12.92	11.74	5.14	3.77	8.39	58.78	51.97	55.37
Initial RL		45.87	36.91	22.00	23.99	32.19	45.87	36.91	22.00	23.99	32.19	72.91	58.51	65.71
IR-OPD-RL														
1	G-OPD	45.23	37.73	22.10	23.73	32.20	45.23	37.73	22.10	23.73	32.20	72.10	62.48	67.29
	RL	50.19	38.82	24.32	27.38	35.18	53.82	39.55	24.15	27.20	36.18	74.40	62.81	68.61
2	G-OPD	51.72	39.57	24.13	28.35	35.94	54.65	39.69	24.08	28.39	36.70	76.24	63.21	69.72
	RL	**53.66**	40.50	24.15	28.14	36.61	54.44	40.12	**24.41**	28.28	36.81	**78.10**	62.90	70.50
3	G-OPD	53.02	40.36	**24.64**	28.30	36.58	55.02	39.31	23.42	27.40	36.28	77.20	62.90	70.05
	RL	53.65	**42.03**	24.27	**28.70**	**37.16**	**55.63**	**41.19**	22.65	**28.48**	**37.00**	77.70	**66.30**	**72.00**
Qwen3-4B-Non-Thinking														
SFT Anchor		22.40	20.10	11.25	8.12	15.47	22.40	20.10	11.25	8.12	15.47	74.40	63.20	68.80
Initial RL		59.06	54.83	32.26	36.81	45.74	59.06	54.83	32.26	36.81	45.74	85.18	70.71	77.95
IR-OPD-RL														
1	G-OPD	61.72	54.62	32.93	37.80	46.77	61.72	54.62	32.93	37.80	46.77	84.29	69.19	76.74
	RL	65.38	58.94	36.75	40.31	50.35	66.49	59.86	36.86	44.60	51.95	87.28	71.02	79.15
2	G-OPD	65.19	58.37	38.21	41.28	50.76	66.44	**60.05**	38.04	45.07	52.40	89.47	71.33	80.40
	RL	68.07	60.68	39.98	43.94	53.17	68.92	59.86	38.14	45.94	53.22	89.70	**72.50**	81.10
3	G-OPD	68.23	60.49	**41.63**	44.13	53.62	68.47	57.88	38.96	**47.07**	53.09	89.70	**72.50**	81.10
	RL	**68.65**	**62.08**	39.79	**45.36**	**53.97**	**69.22**	58.33	**39.84**	46.30	**53.42**	**90.00**	**72.50**	**81.25**

Table 2: **Comparison with representative state-of-the-art post-training methods on Qwen3-4B-Non-Thinking.** All values are accuracies (%). Math Avg. averages AIME24, AIME25, and HMMT25 (Feb./Nov.); Code Avg. averages HumanEval+ and MBPP+. IR-OPD-RL reports the policy after three rounds. Best results are in bold.

	Math-57K					Code-25K		
Method	**A24**	**A25**	**H25-Feb.**	**H25-Nov.**	**Avg.**	**HE+**	**MBPP+**	**Avg.**
RL teacher	59.06	54.83	32.26	36.81	45.74	85.18	70.71	77.95
SFT student	22.40	20.10	11.25	8.12	15.47	74.40	63.20	68.80
ExPO (Zheng et al., 2025)	58.70	55.20	32.40	37.00	45.80	84.80	70.20	77.50
OPD (Agarwal et al., 2024)	60.70	55.00	32.40	37.90	46.50	85.20	69.90	77.55
ExOPD (Yang et al., 2026)	62.70	56.10	33.90	39.30	48.00	86.90	70.70	78.80
IR-OPD-RL (ours)	**69.22**	**58.33**	**39.84**	**46.30**	**53.42**	**90.00**	**72.50**	**81.25**

5.3 IR-OPD–RL Outperforms Continued RL under Matched Compute

To determine whether the gains of IR-OPD-RL arise solely from additional training compute, we compare its iterative checkpoints with continued-RL baselines at matched cumulative-compute budgets in both mathematical reasoning and code generation. Figure 3 reports the domain-average accuracy on Qwen3-1.7B.

On Math-57K, IR-OPD-RL consistently outperforms continued RL, with margins of $+1.84$, $+2.10$, and $+2.35$ points over the three rounds. Continued RL remains between 34.34 and 34.71, whereas IR-OPD-RL improves from 36.18 to 37.00. On Code-25K, the two methods are comparable after the first round (68.61 vs. 68.65), but IR-OPD-RL subsequently reaches 70.50 and 72.00, exceeding continued RL by 1.65 and 3.70 points. Thus, the advantage of IR-OPD-RL is stable from the first round on Math-57K and emerges over later rounds on Code-25K, indicating benefits beyond allocating the same compute to continued RL.

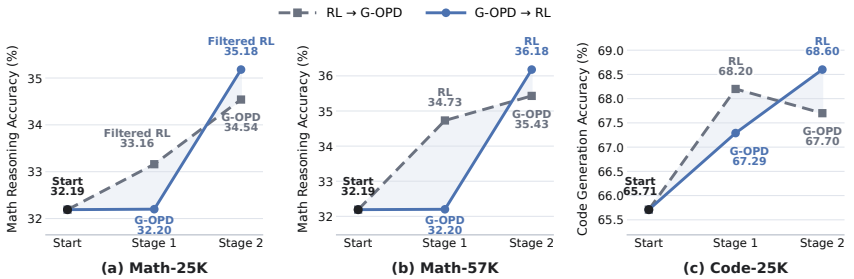


Figure 4: **Single-round order ablation on Qwen3-1.7B.** Starting from the same teacher, G-OPD → RL and RL → G-OPD apply identical stages in reverse order on Math-25K, Math-57K, and Code-25K. RL runs for three epochs; Math-25K RL uses Pass-1–7. Axes are independently scaled, and values are accuracies (%).

Table 3: **Compute-matched one-shot and iterative training on Qwen3-1.7B.** The one-shot schedule (long G-OPD → long GRPO) and IR-OPD-RL are matched for total training compute, reported in 8-NPU node-hours.

Dataset	Training schedule	Total compute	Domain Avg.
Code-25K	Long G-OPD → Long RL	2899	69.30
	IR-OPD–RL		**70.50**
Math-57K	Long G-OPD → Long RL	5867	36.31
	IR-OPD–RL		**36.81**

Table 4: **Effect of student initialization on Qwen3-1.7B Math-25K.** The first-round RL policy $T_1 = 35.18$ is fixed as the G-OPD teacher, while only the student initialization varies.

Student initialization	After G-OPD	After RL	Δ_{RL}
R1 G-OPD	34.84	34.20	−0.64
Initial RL	35.17	34.84	−0.33
Fresh SFT anchor B	**35.94**	**36.61**	**+0.67**

5.4 Ablation Studies

Effect of the ordering of G-OPD and RL. We isolate the ordering effect within a single iteration on Qwen3-1.7B, covering both mathematical reasoning and code generation. Starting from the same initial RL teacher, we compare distillation followed by task-specific RL (G-OPD → RL) with the reversed order (RL → G-OPD), while holding the two training stages fixed. Each RL stage is trained for three epochs; for Math-25K, RL uses the pass-1–7 filtered subset. As shown in Figure 4, applying G-OPD before RL consistently produces the stronger endpoint. On Math-25K, G-OPD → RL reaches 35.18, compared with 34.54 for RL → G-OPD(+0.64%). The same ordering benefit appears on Math-57K, where the two routes reach 36.18 and 35.43, respectively (+0.75%), and on Code-25K, where they reach 68.60 and 67.70 (+0.90%). Thus, the endpoint advantage is reproduced across both mathematical reasoning and code generation, as well as across two math-data settings.

Two IR-OPD-RL rounds versus a compute-matched long G-OPD→RL schedule. We compare two rounds of IR-OPD-RL with a compute-matched one-shot schedule consisting of a single long G-OPD stage followed by a single long RL stage. Both schedules start from the same Qwen3-1.7B policy and use the same total training compute. As shown in Table 3, IR-OPD-RL reaches 70.50 on Code-25K and 36.81 on Math-57K, compared with 69.30 and 36.31 for the one-shot schedule. Repeated alternation therefore improves the compute-matched endpoints by 1.20 and 0.50 points, respectively. Together with the single-round order ablation, these results favor placing G-OPD before RL and refreshing this handoff across rounds.

Table 5: Comparison of RL-only controls and IR-OPD–RL on Qwen3-1.7B Math-25K. All routes use the incoming round-1 policy $T_1 = 35.18$ and matched compute. Re-filtering refreshes the Pass-1–7 subset. Bold denotes the best RL-only and overall results.

Route	RL initialization	Optimizer	Math Avg.
Continued RL	T_1	inherited	34.68
Re-filtered RL	T_1	inherited	**35.41**
Optimizer restart	T_1	reset	34.25
IR-OPD–RL	$S_2 = \mathrm{G\text{-}OPD}(B, T_1)$	reset	**36.61**

Effect of student initialization on G-OPD and subsequent RL To isolate the role of reset, we conduct a controlled experiment on Qwen3-1.7B with Math-25K. We fix the first-round RL policy

Table 6: Efficiency of Pass-1–7 filtering on Math. (a) RL-stage cost and final accuracy on Qwen3-1.7B using Full-25K, a size-matched random subset, and Pass-1–7 filtering. (b) Accuracy and cumulative training cost after three G-OPD-RL rounds. All costs are reported in 8-NPU node-hours.

(a) Single-stage cost and accuracy

Metric	Full-25K	Random	Pass-1–7
Retained prompts	25,000	7,361	7,361
RL Training cost	1,320.00	381.93	381.93
Filter cost	0	0	116.80
Total cost	1,320.00	381.93	498.73
Math Avg.	34.19	32.76	**35.18**

(b) Three-round cumulative efficiency

Model	RL data	Math Avg.	Cumulative compute
Qwen3-1.7B	Full-57K	37.00	3,571.90
	Pass-1–7	**37.16**	**1,321.63**
Qwen3-4B	Full-57K	53.42	3,743.13
	Pass-1–7	**53.97**	**1,473.07**

$T_1 = 35.18$ as the teacher for the next G-OPD stage and vary only the student initialization: the preceding R1 G-OPD checkpoint, the initial RL policy T_0, or the fixed SFT anchor B. All runs use the same teacher, G-OPD data, update budget, and short-RL configuration. As shown in Table 4, initialization from B yields the strongest post-G-OPD checkpoint at 35.94, exceeding the other two initializations by 1.10 and 0.77 points. It is also the only setting with a positive subsequent RL gain ($+0.67$ points), reaching 36.61 and finishing 2.41 and 1.77 points above the alternatives. Thus, in this controlled setting, resetting to the SFT anchor benefits both the immediate G-OPD endpoint and subsequent RL improvement.

Alternative routes for further policy improvement in a low-gain regime. We compare four strategies for further improving the same low-gain policy, the Qwen3-1.7B Math-25K round-1 RL checkpoint $T_1 = 35.18$: continued RL, restarting the optimizer before RL, continuing RL with refreshed Pass-1–7 filtering, and launching a new reset G-OPD–RL round. The first three routes retain the parameters of T_1, whereas IR-OPD–RL transfers its behavior to a student initialized from the fixed SFT anchor B before renewed RL. Table 5 shows that optimizer restart and continued RL reach 34.25 and 34.68, respectively, both below T_1. Re-filtering produces the strongest RL-only result at 35.41, a gain of 0.23%. In comparison, IR-OPD–RL reaches 36.61, improving over T_1 by 1.43% and exceeding the best RL-only intervention by 1.20%. These results suggest that transferring policy behavior to a student reset from the SFT anchor provides a stronger starting point than further modifying the same RL trajectory.

5.5 Pass-1–7 Filtering Improves IR-OPD-RL Compute Efficiency

Under binary rewards, pass-0 and pass-8 prompts have no within-group reward variation and therefore provide no direct group-relative policy-gradient signal. For the designated Math-25K RL stage, we use the post-G-OPD policy to sample eight responses per prompt and retain

$$c(x) = \sum_{i=1}^{8} \mathbb{I}[r(x, y_i) = 1], \qquad \mathcal{D}_{\text{p1-7}} = \{x \in \mathcal{D}_{25\text{K}} : 1 \le c(x) \le 7\}. \tag{4}$$

This criterion is checkpoint-specific and does not represent intrinsic prompt difficulty. Starting from the same post-G-OPD checkpoint, Pass-1–7 reaches 35.18 accuracy, compared with 34.19 for Full-25K and 32.76 for a size-matched random subset. Measured in 8-NPU node-hours, its total cost is 498.73, including 116.80 for filter construction, which is 62.2% below the $1{,}320.00$ full-data cost (Table 6(a)). Applying Pass-1–7 filtering only to the first-round RL stage, the resulting three-round IR-OPD-RL run reaches 37.16, compared with 37.00 for Full-25K on Qwen3-1.7B, and 53.97, compared with 53.42, on Qwen3-4B. Including filter construction, this reduces cumulative compute by 63.0% and 60.6%, respectively (Table 6(b)). Thus, checkpoint-specific filtering substantially accelerates IR-OPD-RL without sacrificing final performance.

6 Conclusion

We introduce IR-OPD-RL, which repeatedly transfers the latest RL policy through reward-extrapolated G-OPD to a student reset from a fixed SFT anchor, followed by renewed GRPO. Across two model scales and six model–data configurations, IR-OPD-RL improves the initial RL teachers by 3.30–8.23 points. Under matched compute on Qwen3-1.7B, it outperforms continued RL by 2.35% on Math-57K and by up to 3.70% on Code-25K. Ordering and trajectory analyses further show that G-OPD is most effective before RL and that its immediate accuracy does not fully predict subsequent RL gains.

AI USE STATEMENT

Generative AI was used to assist with manuscript organization, translation, and language editing. The authors reviewed and verified all AI-assisted text and take full responsibility for the paper.

REPRODUCIBILITY STATEMENT

We describe the model checkpoints, data construction, reward functions, evaluation protocol, and compute accounting in the experimental setup. Algorithm 1 specifies the complete IR-OPD-RL procedure, while Appendix A.1 reports the optimization, sampling, rollout-correction, and distributed-training configurations. All comparisons use the same evaluation harness and explicitly report any data filtering or compute matching.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.13649.

AI-MO. Aimo validation aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024. Hugging Face dataset.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *arXiv preprint arXiv:1705.08439*, 2017. URL https://arxiv.org/abs/1705.08439.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, 2026. URL https://arxiv.org/abs/2505.23281.

Yutong Chen, Jiandong Gao, and Ji Wu. Towards revealing the effectiveness of small-scale fine-tuning in r1-style reinforcement learning. *arXiv preprint arXiv:2505.17988*, 2025. URL https://arxiv.org/abs/2505.17988.

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, 2025. URL https://arxiv.org/abs/2502.01456.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Nature*, 645:633–638, 2025. URL https://arxiv.org/abs/2501.12948.

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, 2025. URL https://arxiv.org/abs/2504.11456.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2021. URL https://arxiv.org/abs/2006.05826.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Zhuo Jiang. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025a. URL https://richardli.xyz/rl-collapse.

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.

Runze Liu, Jiashun Liu, Xu Wan, Yuqian Fu, and Ling Pan. When rl fails after sft: Rejuvenating model plasticity for robust sft-to-rl handoff. *arXiv preprint arXiv:2606.09932*, 2026. URL https://arxiv.org/abs/2606.09932.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b. URL https://arxiv.org/abs/2503.20783.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026. URL https://arxiv.org/abs/2606.30406.

OpenCompass. Aime 2025 dataset. https://huggingface.co/datasets/opencompass/AIME2025, 2025. Hugging Face dataset.

Miao Rang, Zhenni Bi, Hang Zhou, Kai Han, Xuechun Wang, An Xiao, Xinghao Chen, Yunhe Wang, and Hanting Chen. Near-policy: Accelerating on-policy distillation via asynchronous generation and selective packing, 2026. URL https://arxiv.org/abs/2605.05940.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 15:627–635, 2011. URL https://arxiv.org/abs/1011.0686.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2016. URL https://arxiv.org/abs/1511.06295.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024. URL https://arxiv.org/abs/2410.08146.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2402.03300.

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. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025. URL https://arxiv.org/abs/2506.14245.

Teng Xiao, Yige Yuan, Mingxiao Li, Zhengyu Chen, and Vasant G Honavar. On a connection between imitation learning and rlhf, 2025. URL https://arxiv.org/abs/2503.05079.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-RL: Unleashing LLM reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025. URL https://arxiv.org/abs/2502.14768.

Hongling Xu, Qi Zhu, Heyuan Deng, Jinpeng Li, Lu Hou, Yasheng Wang, Lifeng Shang, Ruifeng Xu, and Fei Mi. Kdrl: Post-training reasoning llms via unified knowledge distillation and reinforcement learning. *arXiv preprint arXiv:2506.02208*, 2025. URL https://arxiv.org/abs/2506.02208.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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.

Qinglin Ye, Zhiyuan Gu, Jingjie Xia, Yiheng Zhang, Kaiyan Zhao, Shunchao Zheng, Yuhang Mu, Wenchao Du, and Yiming Wang. Opdsearch+: On-policy distillation with rl refinement for search-augmented reasoning. *arXiv preprint arXiv:2608.24310*, 2026. URL https://arxiv.org/abs/2608.24310.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. URL https://arxiv.org/abs/2504.13837.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL https://arxiv.org/abs/2203.14465.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Model extrapolation expedites alignment, 2025. URL https://arxiv.org/abs/2404.16792.

A APPENDIX

A.1 TRAINING AND IMPLEMENTATION DETAILS

Table 7 reports the default configurations used for each G-OPD and GRPO stage. G-OPD samples one response for each of 1,024 prompts, whereas GRPO samples a group of eight responses for each of 128 prompts. Thus, both methods process 1,024 sampled sequences per update. Every stage is trained for three epochs. Parameters not listed in the table use the default settings of `verl`.

Table 7: **Optimization and rollout hyperparameters.** Values are shared across model scales and domains unless stated otherwise.

Hyperparameter	**G-OPD**	**GRPO**
Training objective	Reverse-KL advantage	Group-relative outcome reward
Reward-scaling coefficient λ	1.25	–
Prompt batch size	1,024	128
Rollouts per prompt	1	8
Sampled sequences per update	1,024	1,024
Optimizer	AdamW	AdamW
Learning rate	1×10^{-5}	1×10^{-6}
Weight decay	0.01	0.01
Learning-rate warmup ratio	0	0
Optimizer epochs per batch	1	1
Mini-batch size	1,024	128
Micro-batch size per NPU	1	1
Training epochs per stage	3	3
Maximum prompt length	2,048	2,048
Maximum response length	16,384	16,384
Rollout temperature / top-p	1.0 / 1.0	1.0 / 1.0
Entropy coefficient	0	0
KL penalty coefficient	0	0
Rollout correction	Token-level IS	Token-level IS
IS-ratio threshold	5.0	5.0
Data shuffling / seed	Yes / 42	Yes / 42
Thinking mode	Disabled	Disabled

Rollouts are generated with `vLLM`. The default distributed configuration uses eight Ascend 910B NPUs per node, tensor parallelism of 4, and Ulysses sequence parallelism of 2. We enable gradient checkpointing and remove padding, cap the number of batched tokens at 32,768, and set rollout device-memory utilization to 0.3. Prompts longer than 2,048 tokens are filtered before training; truncation is otherwise disabled.

Teacher schedule. In the first IR-OPD-RL round, G-OPD uses the available domain-specific RL teachers under a multi-teacher configuration. In the second and third rounds, the GRPO policy produced by the preceding round serves as the sole teacher for the corresponding domain. Thus, multi-teacher supervision is used only to initialize the iterative process; subsequent rounds follow the single-teacher recurrence in Eq. 1, with the teacher refreshed after each GRPO stage.