DECOUPLING EXPLORATION FROM OPTIMIZATION IN RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern language models undergo reinforcement learning with verifiable rewards (RLVR) on top of already-trained checkpoints. A key promise of RLVR is the discovery of new reasoning strategies. In principle, a model can sample novel ideas absent from its prior training data. In practice, however, augmenting RLVR with strong novelty incentives has seen limited success and can degrade model quality. Because verifiable rewards supervise only a narrow slice of the model's knowledge and behavior, such degradations are difficult to recover from. Instead, we decouple exploration from optimization in a framework we call Exploration-Distillation (ExpDis). We train one or more *explorer* policies with a novelty bonus in the reward, filter their trajectories for correctness and quality, and distill them into a separate *student* policy. The student policy is then trained without a novelty bonus. We repeat the above procedure for several rounds, alternating between exploration and optimization. This decoupling allows us to aggressively scale exploration without degrading the student policy. Across seven mathematical reasoning benchmarks and two model families, ExpDis outperforms DAPO at the same wall-clock budget. Moreover, we observe improved pass@k scaling, indicating that ExpDis produces models that generate more diverse correct solutions.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become the dominant recipe for improving the reasoning abilities of language models (Lambert et al., 2024; Guo et al., 2025; Shao et al., 2024; Liu et al., 2024; Yu et al., 2026). RLVR samples solutions to problems, scores each solution with a programmatic verifier, and shifts probability toward those that pass. By iteratively sampling new solutions and reinforcing successful ones, RLVR can in principle discover and then reinforce new solutions that were absent from the model's training data. In practice, however, it can only reinforce what the model samples, so discovery depends on exploration.

A natural way to boost exploration is to add an explicit novelty bonus to the reward, following a long line of intrinsic-motivation methods in RL (Pathak et al., 2017; Bellemare et al., 2016; Burda et al., 2018; Eysenbach et al., 2018; Ecoffet et al., 2021). When the model explores new solution strategies, some of them may succeed but often most of them fail, leading to a temporarily degraded model as we now are left with a model that mostly generates failed solutions. Though our initial policy is a pretrained LLM with vast knowledge including virtually all of Wikipedia, the reward supervises only a tiny slice of its knowledge. This means an LLM could forget most of Wikipedia and still achieve high reward. The reward therefore cannot detect damage that exploration causes outside that slice, let alone repair such degradations. We argue that this failure is not a property of novelty as a signal, but of where the novelty bonus is applied.

We decouple exploration and optimization by giving them separate model instances, both trained from the same base checkpoint. An *explorer policy* explores aggressively under a novelty bonus whose weight λ sets how strongly it is pushed toward unfamiliar solutions relative to correctness. A separate *student policy* never sees a novelty reward. It learns from the explorer only through filtered trajectories, and is then trained with standard RLVR. The two models communicate through filtered trajectories rather than parameters, so the student policy absorbs the explorer's discoveries without inheriting its degradations induced by exploration. We call this framework *Exploration-Distillation*.

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.

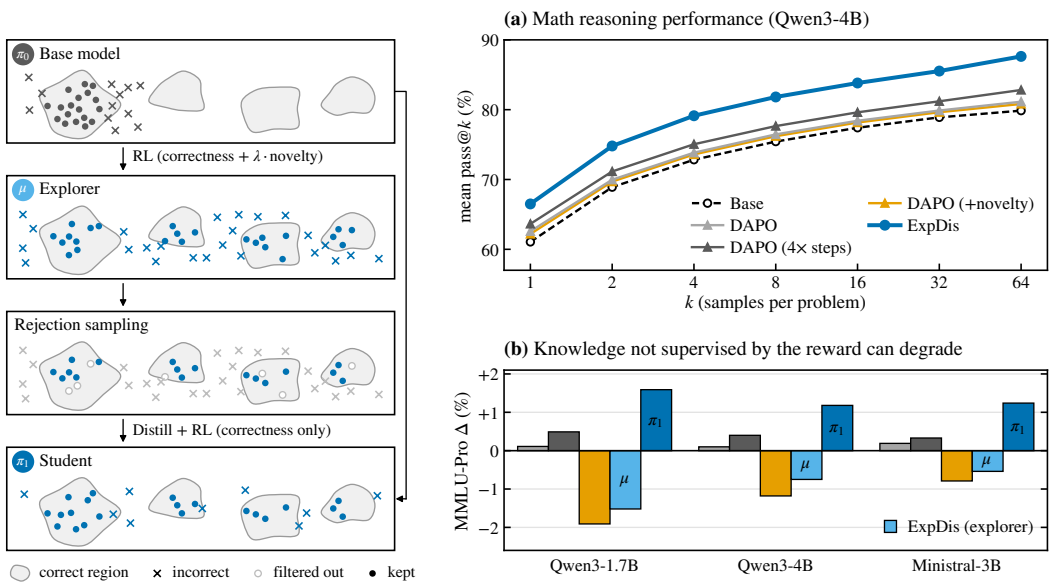


Figure 1: Rather than coupling exploration and optimization in a single on-policy RLVR run, ExpDis assigns these roles to separate policies. We first train an explorer μ with a novelty bonus in its reward, then distill a diverse pool of its reasoning traces to cold start a student π_1, and finally train the student with standard RLVR using only a correctness reward. **(a)** ExpDis outperforms DAPO in both average accuracy and pass@k, even when DAPO is trained for $4\times$ as many RL steps. **(b)** Adding an RND novelty bonus directly to the DAPO reward lowers MMLU-Pro accuracy below the base model, whereas ExpDis, which applies the bonus only to the explorer policy, does not. Because the reward supervises only a narrow slice of the model's knowledge (here, mathematical reasoning), it does not prevent this degradation.

Decoupling makes exploration safe to scale aggressively, since the student policy never receives the novelty bonus and the filter keeps failed trajectories out of its training data. We can scale *breadth* by running several explorers in parallel and distilling their pooled trajectories into one student policy. We can also scale *depth* by repeating the pipeline for multiple rounds that each start from the previous student. We scale both of these axes independently and find that their composition enables sustained performance gains even when all experiments are wall-clock matched.

We evaluate ExpDis on mathematical reasoning benchmarks with Qwen3-1.7B, Qwen3-4B, and Ministral-3-3B-Instruct-2512 as base models. At the same amount of compute, ExpDis outperforms DAPO in both pass@1 and pass@64. Furthermore, spreading this fixed budget over multiple parallel explorers and several rounds of the ExpDis procedure yields further gains. Adding a novelty bonus directly to DAPO, in contrast, increases diversity but degrades performance on MMLU-Pro, a general knowledge benchmark. In ExpDis, the explorer can be pushed to far higher diversity without harming the student, which improves in both diversity and accuracy and avoids degradation.

In principle, ExpDis is compatible with any exploration mechanism, since the student learns only from the explorer's filtered trajectories. Since the explorer is never deployed, it can also use mechanisms too aggressive to apply to the deployed model directly. Exploration methods that would destabilize standard RLVR may therefore become viable under ExpDis.

2 Related Work

Exploration in LLM reasoning. Whether RLVR teaches models new reasoning or only sharpens what they can already do. Shao et al. (2024) observe that RL improves majority-vote accuracy but not pass@K, and Chen et al. (2026) find that base models overtake their RLVR-trained counterparts at large k, suggesting that RLVR narrows the set of problems a model can solve. Others find that RL can expand this set through prolonged training (Liu et al., 2026b), or that it learns new skills by composing existing ones (Yuan et al., 2026). A related line of work studies entropy collapse: policy entropy falls sharply early in training as performance saturates (Cui et al., 2025), and the gains from RLVR come mostly from a small set of high-entropy tokens (Wang et al., 2026b). Many methods counteract

this collapse during training. For example, DAPO (Yu et al., 2026) raises the upper clipping bound so that low-probability tokens with positive advantage can grow faster, whereas He et al. (2025) directly regulate the policy's entropy. Other approaches add exploration or novelty bonuses to the reward (Song et al., 2025; Gao et al., 2025; Tuyls et al., 2026; Li et al., 2025). These methods draw on a long line of exploration bonuses in classical RL, including count-based bonuses (Bellemare et al., 2016), intrinsic curiosity (Pathak et al., 2017), and Random Network Distillation (Burda et al., 2018).

Reasoning traces distillation. Training a policy to imitate a separate process that finds better solutions dates back to expert iteration (Anthony et al., 2017), in which a tree-search expert generates improved moves and a neural apprentice learns to reproduce them. Today, distilling reasoning traces from a stronger model is standard practice: DeepSeek-R1 (Guo et al., 2025) is distilled into smaller open models, and several efforts on how to curate reasoning traces to cold-start effective RL exist (Ye et al., 2025; Guha et al., 2026). When no stronger model is available, a model can instead learn from its own filtered samples, as in STaR (Zelikman et al., 2022), ReST (Gulcehre et al., 2023), and ReST^{EM} (Singh et al., 2023). DeepSeek-R1 fine-tunes its base model on correct traces rejection-sampled from an RL checkpoint before running RL again and multi-teacher on-policy distillation methods (Ma et al., 2026) repeatedly distill RL rollouts into a fresh checkpoint and consolidates separately trained specialist models into one model before a final RL stage. We follow a similar structure, but train our teacher policies from the same model to seek novel solutions.

Decoupling exploration. In classical RL, DEEP (Whitney et al., 2021) and DeRL (Schäfer et al., 2021) train separate exploration and task policies, since bonus-driven exploration can bias or destabilize the task policy, and Mhammedi & Cohan (2026) explore with uncertainty-guided tree search before distilling the discovered trajectories into a policy. For LLMs, R^{2}PO (Wang et al., 2026a) explores with an auxiliary rollout head that is discarded at inference, and in concurrent work, ICE (Dilkes et al., 2026) explores with instruction-conditioned prompts and self-distills correct rollouts into the unconditioned policy. In both, exploration and the deployed model share weights. We instead train the explorer as a separate network and pass only filtered trajectories to the student, such that no exploration gradient reaches the deployed model.

3 METHOD

Exploration-Distillation trains an explorer policy μ with a novelty bonus to generate diverse trajectories, filters them for quality and correctness, and distills the resulting trajectories into a student policy π_1. We present a single iteration below and describe how we scale it in Section 3.1 and Section 3.2. We study the importance of each stage in Appendix E.

Explorer Policy Training. We initialize an explorer policy μ from the base policy π_0 with DAPO (Yu et al., 2026) and augment the reward with a novelty reward $r_{\text{novelty}} \geq 0$.

$$r_{\text{explorer}}(\tau) \;=\; r_{\text{correct}}(\tau) \;+\; \lambda \cdot r_{\text{novelty}}(\tau) \cdot \mathbf{1}[r_{\text{correct}}(\tau) = +1] \;+\; r_{\text{overlong}}(\tau), \tag{1}$$

where $r_{\text{correct}} \in \{-1, +1\}$ and $\lambda \geq 0$. As introduced in DAPO, $r_{\text{overlong}} \in [-1, 0]$ is a soft penalty on completions that approach the length budget. This reward formulation ensures that every correct rollout receives a higher reward than any incorrect one for any value of λ. We find that applying the bonus to all rollouts including incorrect ones leads to training instability. Apart from the novelty term, the explorer is trained exactly like our DAPO baseline, so differences in its behavior come from the bonus alone.

Rejection Sampling. We now need a method for collecting trajectories generated by the explorer policy to be distilled into the student. We must remove any incorrect trajectories, since the point of this rejection sampling step is to insulate the student from any degradations caused to the explorer by novelty pressure. Naively, we could simply take all correct explorer trajectories or a random subset and hand them off, but we find that filtering not only for correctness but also for quality strongly improves performance. A trajectory passes the filter only if the verifier accepts its final `\boxed{}` answer, it terminates within the 32,768-token budget, and it contains no repetition loops. These checks remove the looping and unterminated completions that strong novelty pressure can produce. After this filtering, we keep the shortest resulting trajectory per problem. As a result, each

solved problem contributes a single example, and problems that the explorer solves many times do not dominate the student's training data. We cap the resulting set at 500 trajectories per round. In Appendix E, we ablate our quality filtering procedure.

Student Policy Training. We distill the base policy on the resulting trajectories via supervised finetuning (SFT) with a standard cross-entropy loss resulting in a student policy π_1. We then train the student policy with DAPO and the standard correctness-only reward (i.e., no novelty bonus), which inherits none of the degradation from exploration pressure and insulates the student from the degradations we observe in the explorer. The student starts from the base policy rather than from the explorer, so none of the explorer's weight updates reach it. SFT transfers the explorer's solutions to the student, and the RL stage then trains the student on its own samples, following the common practice of distillation followed by RL (Guo et al., 2025). Both stages are necessary. Distilling filtered traces without the RL stage, or running RL after distilling unfiltered traces, each performs worse than DAPO (Appendix E).

3.1 Parallel Explorer Policies

Rather than train a single explorer policy μ, we can train K independent explorer policies and pool their trajectories before filtering and performing distillation. Training parallel explorers an increase the diversity of trajectories we curate for the student and the increased coverage raises our probability of discovering successful solutions. Each explorer starts from the same policy but uses its own random seed, RND networks, and data order. We merge the trajectories of all K explorers and apply the filter above to the merged pool. We set the novelty bonus weight λ identically for all explorer policies, as we find no benefit from varying them (see Appendix C). Crucially, all explorers can be scaled independently and without additional communication overhead, requiring no additional wall-clock time over a single explorer in our experiment and allow controlled comparisons.

3.2 Iterative Exploration-Distillation

After distilling a student policy, we can use this as a strong initialization for both the next explorer policy and subsequent student policy. Formally: to train π_t, we first train an explorer μ from π_{t-1}, filter the training trajectories, perform distillation on π_{t-1} to produce π_t, and perform standard RLVR.

We partition the training prompts into R non-overlapping shards $\mathcal{D}_1, \ldots, \mathcal{D}_R$ of equal size. As shown in Figure 2, round r draws its training prompts from $\mathcal{D}_r$. Hence, each round pairs a stronger initialization with prompts it has not yet trained on. Rollouts on these prompts have not been sharpened by earlier rounds, so they remain diverse enough for exploration to be informative.

Annealing novelty bonus. Because exploration is split across several rounds, we can set its strength per round through the novelty-bonus weight λ, which we anneal on a fixed schedule from 0.75 to 0.25 over four rounds. This follows the standard practice of decaying exploration as uncertainty shrinks (Sutton et al., 1998). Early rounds, where correct trajectories are still undiscovered, get strong novelty pressure whereas later rounds get low novelty pressure to consolidate what has been found.

4 Experimental Setup

Base models. We evaluate ExpDis on Qwen3-1.7B, Qwen3-4B (Yang et al., 2025), and Ministral-3-3B-Instruct-2512 (Liu et al., 2026a). The Qwen3 models are distilled from larger RL-trained Qwen3 models rather than trained with RL directly (Yang et al., 2025), whereas Ministral-3-3B-Instruct is instruction-tuned and undergoes online preference optimization. The two Qwen3 sizes test whether the gains hold as the model grows, and Ministral tests a second model family with a different post-training recipe. We run the Qwen3 models in thinking mode. Ministral has no thinking mode and runs as a standard instruction-following model. We describe all sampling parameters in Appendix A.

Novelty bonus. In principle, ExpDis is bonus-agnostic and we can simply swap out different novelty bonuses. In our primary experiments, we choose a Random Network Distillation (RND) bonus (Burda et al., 2018) as we find that this formulation is especially effective. Following Burda et al. (2018), we train one small predictor MLP $f_{\theta,\ell}$ per layer to match a frozen random target MLP

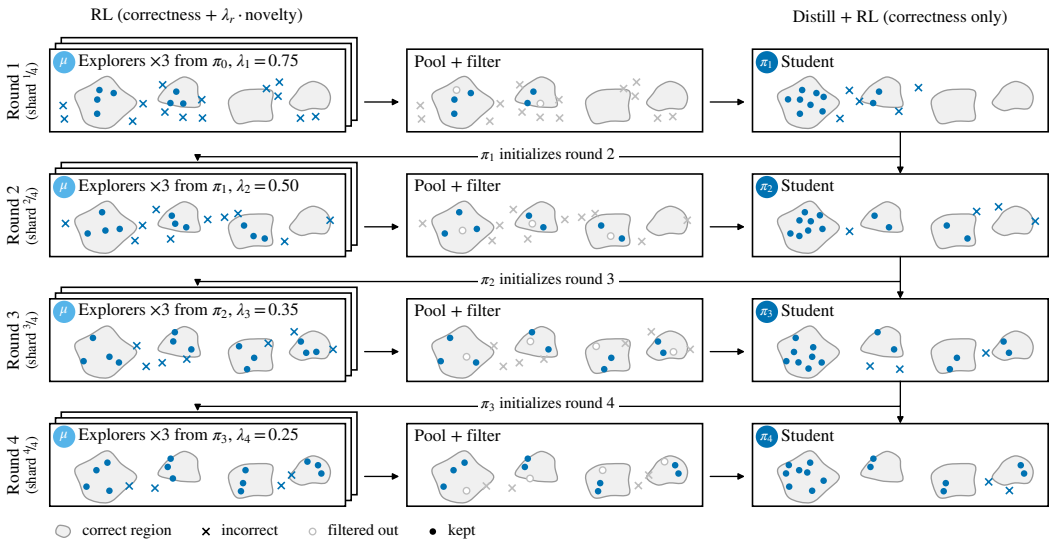


Figure 2: Scaling exploration. We can scale ExpDis along two axes. Within a round, several explorers are trained in parallel with a novelty bonus, and their traces are pooled to distill a single student. Across rounds, that student initializes both the next round's explorers and the next round's student, so every round starts from a stronger model. We study how to allocate a fixed RL budget across explorers and rounds in Section 5.

$f_\ell^\star$ on features extracted from the language model's own hidden states, and use the prediction error as the novelty signal. Computing the bonus from hidden states rather than output tokens makes it depend on how the model represents a solution rather than on its wording. We extract features from a fixed set of intermediate layers $\mathcal{L}$, mean-pool over non-pad tokens, and average across layers:

$$r_{\text{novelty}}(\tau) \;=\; \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}}\frac{1}{\sqrt{d}}\left\|f_\ell^\star\big(h_\ell(\tau)\big) - f_{\theta,\ell}\big(h_\ell(\tau)\big)\right\|_2. \tag{2}$$

Here $h_\ell(\tau)$ is the pooled layer-ℓ feature of trajectory τ and d is the output dimension of the predictor and target networks, so each layer contributes a root-mean-square prediction error. The predictor updates continuously on the same features that drive the bonus, while the target stays frozen for the entire run. In preliminary experiments, features from several intermediate layers gave a cleaner signal than last-layer features alone (layers and network sizes in Appendix A). We ablate the choice of r_{novelty} in Appendix B.1 and observe a consistent improvement over DAPO, even with completely different non-RND forms of novelty bonus.

Reward. We use a standard math verifier for r_{correct}. It extracts the final `\boxed{}` answer, normalizes its formatting, and compares it with the reference answer as a string or as a number. The same verifier scores every benchmark. In Section 5, we sweep the weight λ on r_{novelty} and find that even very large weights leave the student model outperforming the base model. We find that performance roughly peaks at λ=0.5 and conduct all experiments with this.

Training dataset. We sample random prompts without replacement from DAPO-Math-17K (Yu et al., 2026), a collection of 17K mathematical reasoning problems, until reaching a cap of 300 steps (discussed below). DAPO-Math-17K is the training set released with DAPO, our strongest baseline, and all main results use it. At four prompts per step, a 300-step run uses about 1,200 prompts, so no run completes an epoch. We perform additional experiments on the DeepScaleR (Luo et al., 2025) dataset in Appendix E.

Evaluation. We report performance on seven mathematical reasoning benchmarks: AIME24, AIME25, AIME26 (of America, 2026), MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), AMC23 (of America, 2026), Minerva-Math (Lewkowycz et al., 2022), and GSM8K (Cobbe et al., 2021). We sample 64 completions per problem and report the unbiased pass@k estimator for $k = 1 \ldots 64$. Unless noted otherwise, we average over AIME24, AIME25, AIME26, MATH500, and

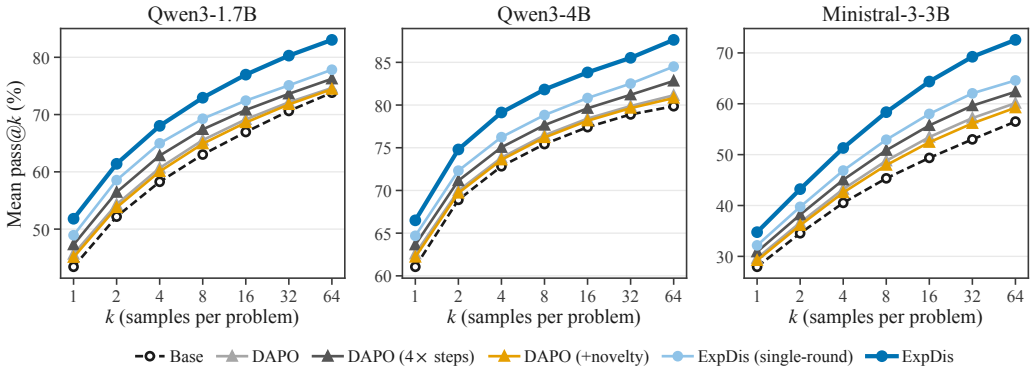


Figure 3: ExpDis outperforms all baselines under the same RL compute budget and DAPO at 4× the compute. By partitioning a fixed amount of compute into alternating rounds of exploration and optimization (as described in Section 3.2), we surpass a single-round of ExpDis. We report full benchmark results in Appendix B.

Minerva-Math. AIME26 was released after all three base models, so it cannot appear in their training data. Pass@1 measures the accuracy of a single sample, while pass@k at large k measures whether a model solves a problem in any of k attempts. Prior work finds that RLVR can raise pass@1 while lowering pass@k at large k (Chen et al., 2026), so we report both. We additionally study the model's degradation on the MMLU-Pro (Wang et al., 2024a) benchmark as a proxy for knowledge that is not supervised by our training datasets.

Baselines. We compare against three RLVR baselines with a standard correctness-only reward: GRPO (Shao et al., 2024), Dr. GRPO (Liu et al., 2024), and DAPO (Yu et al., 2026). GRPO computes each completion's advantage relative to the other completions for the same prompt. Dr. GRPO removes the length and standard-deviation normalizations that bias GRPO's updates, and DAPO adds decoupled clipping, dynamic sampling, a token-level loss, and overlong reward shaping. We find empirically that DAPO is the strongest of these baselines and design two extensions: (1) adding a novelty bonus to the reward with λ=0.5 to isolate the effect of decoupling exploration, and (2) running DAPO for 4× as many steps. ExpDis also uses the DAPO update for both the explorer and the student, so DAPO is the single-policy counterpart of our method. The first extension uses the same reward as our explorer, so the bonus is unchanged but is applied to the policy we evaluate. The second tests whether additional single-policy training closes the gap to ExpDis.

Compute matching. In all experiments, we performed 300 steps of RL, with 4 prompts per step, each having a group size of 16 and a 32,768 max generation limit. We also study the effect of training DAPO for 4x the number of steps to see how prolonged training impacts performance (Liu et al., 2026b).

ExpDis splits these updates between the two policies: 200 for the explorer and 100 for the student. When we scale across explorers or rounds, each share is divided evenly among them, while we keep the SFT stage of the student policy two epochs on at most 500 filtered explorer trajectories per round. We match methods by RL updates so that any gain from ExpDis comes from how it spends the budget rather than from additional training. The short SFT stage is not counted in this budget. Parallel explorers train at the same time, so all configurations take the same wall-clock time (Table 2).

5 Results

Across all three models, ExpDis outperforms DAPO in mean accuracy on mathematical reasoning benchmarks (Figure 3). A single round of ExpDis already outperforms DAPO trained for 4× as many steps. By spreading the same compute over several rounds of alternating exploration and optimization (Section 3.2), we observe further gains. All ExpDis runs outperform this prolonged DAPO run in both average performance and pass@k up to 64 generations per problem.

The single-round comparison is the most direct test of decoupling. It uses the same 300 RL updates as DAPO, split between the explorer and the student, and it outperforms DAPO trained with four times

as many updates, so additional single-policy training does not account for the gain. The advantage of ExpDis also holds at every k we evaluate. The student therefore does not trade coverage for single-sample accuracy, the trade-off that prior work reports for RLVR (Chen et al., 2026).

What each stage contributes. Our ablations on Qwen3-1.7B isolate the parts of the pipeline (Appendix E). Distilling unfiltered explorer traces and then running RL performs no better than DAPO, and distilling filtered traces without the RL stage also falls short of DAPO. Only the combination of the quality filter and the student's RL stage produces the gain, so ExpDis does not reduce to data curation or to additional RL. The choice of novelty signal matters less. Replacing RND with a kNN-based or an elliptical novelty bonus (Tuyls et al., 2026) lowers mean accuracy by at most 1.6 points, and both alternatives still outperform DAPO (Appendix B.1). Across rounds, annealing λ from 0.75 to 0.25 outperforms a fixed λ=0.5 at every round.

Scaling exploration. Decoupling lets us scale exploration along two axes: running several explorers in parallel within a round and repeating the ExpDis procedure over several rounds. In Figure 4, we vary both on Qwen3-1.7B while holding the total budget fixed at 300 RL updates. We observe that depth contributes more to pass@1 than breadth, though a combination of both improves pass@k over either. One interpretation is that the two axes play different roles: each additional round reinforces a progressively stronger student, whereas additional explorers widen the set of problems with correct trajectories that get distilled into the student.

At our budget, gains saturate and begin to show signs of regression at three explorers and four rounds. We expect this optimum to shift as the budget grows: with more compute budget, larger models, and more training data, each round and explorer can train for longer, and we expect the optimal number of rounds and explorers to increase accordingly. Breadth is particularly cheap to scale, since explorers run in parallel and add no wall-clock time.

Diversity analysis. RLVR is known to collapse entropy and sharpen the sampling distribution as training proceeds (Cui et al., 2025; Chen et al., 2026). Counteracting this within a single model requires a delicate balance, since the same model must both explore diverse reasoning strategies and be optimized for correctness. Adding a novelty bonus to DAPO illustrates this trade-off: it increases both lexical and answer diversity relative to standard DAPO, yet slightly reduces mathematical reasoning performance. Diversity and performance therefore need not move together; a model may, for example, learn to rephrase the same solution strategy, thereby increasing lexical diversity or entropy without improving pass@k.

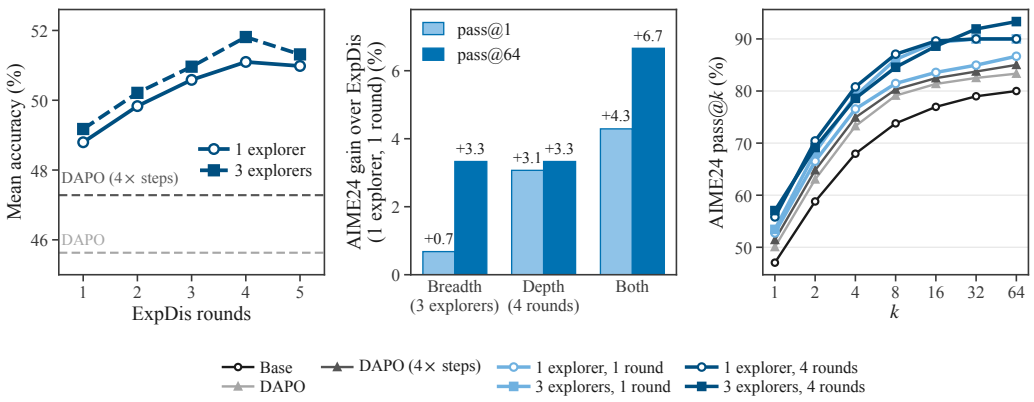


Figure 4: We vary the number of explorers and rounds under a fixed compute budget, and find complementary impacts of each scaling axis. All results are reported on Qwen3-1.7B.

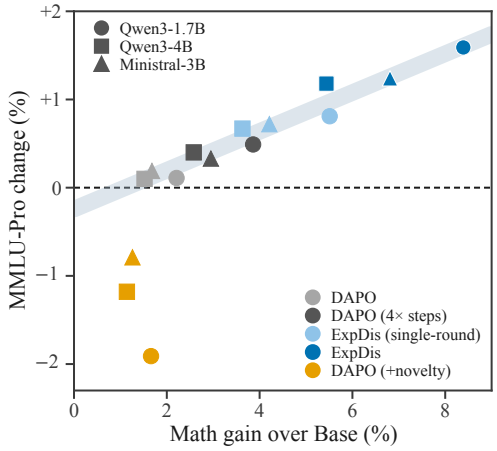
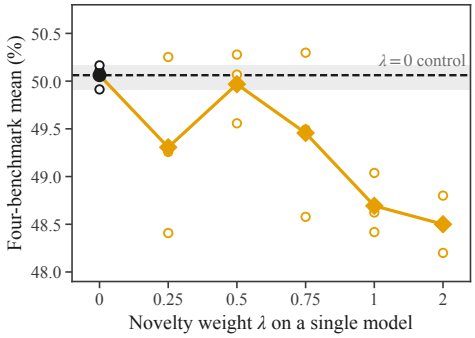


Figure 5: Math gain versus change in MMLU-Pro accuracy relative to the base model. The shaded band is a linear fit to all runs without a direct novelty bonus.

Novelty bonus impact on math reasoning. In the DAPO (+novelty) baseline, we augment the standard correctness-only reward with a novelty bonus as in equation 1. We study how the method performs with various novelty bonus weights λ from 0.25 to 2. We perform three training runs over different seeds and find that, with the RND novelty bonus, the mean mathematical reasoning performance degrades in comparison to DAPO.



As shown in Figure 6, ExpDis improves on both diversity and performance at the same λ. By decoupling the explorer from the student, we can push the explorer toward far greater diversity than a single model could tolerate. We observe that increasing λ steadily raises the explorer's lexical diversity, but only part of it carries over to the student. Instead, the student's diversity and accuracy both peak at moderate λ and decline as the explorer is pushed further. Across rounds of ExpDis, the student's token entropy and accuracy increase together, indicating that the student becomes more diverse without degenerating.

We additionally study the impact that sweeping the novelty bonus weight (λ) on the teacher policy has on the downstream student policy in a single-round ExpDis setup (Figure 6). Though any amount of novelty bonus hurt the teacher policy, even at an high weights (such as λ=2) where pass@k regresses, the downstream student policy does not inherit this degradation.

Novelty bonus degradation. A verifiable reward supervises only a narrow slice of a model's capabilities, so degradation elsewhere can go unnoticed during training. To measure this, we evaluate each model on MMLU-Pro (Wang et al., 2024b), a broad knowledge benchmark (Figure 5). Every other method improves on MMLU-Pro along with math, likely because MMLU-Pro includes mathematical subjects. In contrast, DAPO with the RND novelty bonus degrades below the base model on MMLU-Pro despite also improving on math, suggesting that the policy loses knowledge outside the domain its reward supervises, and that this loss is larger than the net drop shows. While these drops are modest at our scale, we speculate that larger-scale RLVR runs that apply exploration bonuses directly to the trained model may accumulate substantially larger degradations over time, whereas decoupling avoids such a risk.

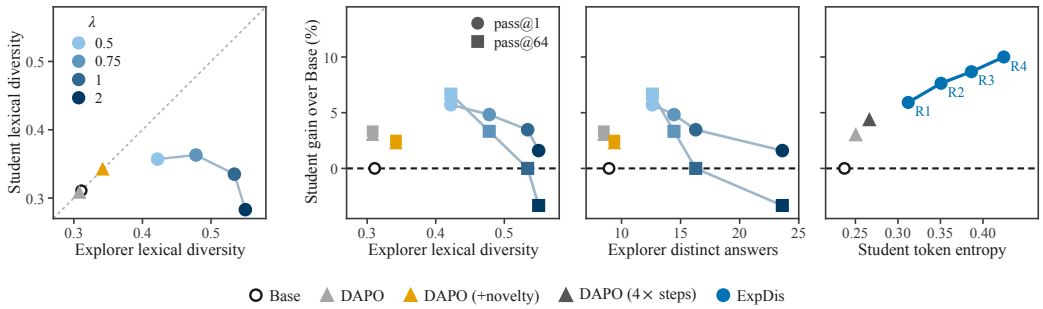


Figure 6: On Qwen3-1.7B, we find that we can push the diversity of the teacher policy without corrupting the downstream student policy. We define each diversity metric in Appendix D.

6 Discussion

In standard RLVR, exploration is limited by the need to remain correct. ExpDis removes this constraint by moving exploration into a separate policy. Because the explorer is never deployed, its objective no longer has to stay close to correctness. It can be rewarded for novelty, pushed to extremes, or discarded after a single round, with only filtered reasoning traces passing to the student.

Our results suggest that decoupling exploration from optimization improves RLVR even at a small budget, and we expect its importance to grow as RL is scaled. As models improve, the problems worth training on are increasingly those where correct solutions are rare and a model optimized for correctness may not be able to sample them. This is especially true beyond mathematical reasoning, in long-horizon agentic tasks, theorem proving, and open-ended scientific problems. Decoupling lets explorers be specialized and aggressive.

In our work, we use a single novelty bonus and explorers that share the student's architecture. In an ExpDis procedure, explorers could use different novelty signals or sampling strategies to specialize to different parts of the problem distribution. Because an explorer only has to produce useful traces, the best exploration method under ExpDis may look quite different from the best one under standard RLVR. We speculate that exploration methods previously too destabilizing to apply directly to a trained model may become usable in this setting.

AI USE STATEMENT

In this work, we used generative AI tools for tasks with required disclosure. AI coding agents helped write and run the experiment code, and they helped draft and tighten the text. We have not used generative AI tools to set the research questions, and data annotation, translation and qualitative analysis are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure, to find and verify citations and to check the paper for consistency. We have reviewed all AI-assisted work. The authors set the research questions, checked the code, results, proofs and citations, and edited the text, and every citation was checked against Crossref, arXiv or OpenAlex. 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 trains and evaluates publicly released language models on public mathematical reasoning datasets and benchmarks. It involves no human subjects and no personal data. One of our findings bears on safe practice: adding a novelty bonus to the reward of the policy being trained can lower its accuracy on MMLU-Pro, outside the domain the reward supervises, and a verifiable reward does not detect such losses. We therefore encourage work that applies exploration bonuses in RLVR to also measure performance outside the rewarded domain.

REPRODUCIBILITY STATEMENT

Section 3 specifies the ExpDis procedure, including the explorer reward, the filtering rules for each distillation set, and the student's training. Section 4 describes the novelty bonus, models, data, evaluation protocol, and baselines. Appendix A lists all hyperparameters (Table 1), the allocation of RL updates across explorers, rounds, and the student for every configuration (Table 2), the hardware, the sampling parameters for training and evaluation (Table 3), and the pass@k estimator. All models, datasets, and benchmarks we use are publicly available. We will release our code and all trained checkpoints.

REFERENCES

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Jim Dilkes, Vahid Yazdanpanah, and Sebastian Stein. Instruction-conditioned exploration with asymmetric reinforcement learning and self-distillation. *arXiv preprint arXiv:2608.02087*, 2026.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Maolin Wang, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing llm reasoning with intrinsic motivation guided exploration. *arXiv preprint arXiv:2505.17621*, 2025.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 108059–108130, 2026.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

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 series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e4f51680, 2025. Notion Blog.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026a.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *Advances in Neural Information Processing Systems*, 38:17998–18031, 2026b.

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, 2025. *URL https://arxiv. org/abs/2503.20783*, 1, 2024.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

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

Zakaria Mhammedi and James Cohan. Decoupling exploration and policy optimization: Uncertainty guided tree search for hard exploration. *arXiv preprint arXiv:2603.22273*, 2026.

Mathematical Association of America. American invitational mathematics examination (aime), 2026.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Lukas Schäfer, Filippos Christianos, Josiah P Hanna, and Stefano V Albrecht. Decoupled reinforcement learning to stabilise intrinsically-motivated exploration. *arXiv preprint arXiv:2107.08966*, 2021.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning. *arXiv preprint arXiv:2509.06941*, 2025.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Jens Tuyls, Dylan Foster, Akshay Krishnamurthy, and Jordan Ash. Representation-based exploration for language models: From test-time to post-training. In *International Conference on Learning Representations*, volume 2026, pp. 120621–120645, 2026.

Jingchu Wang, Bingbing Xu, Yige Yuan, Bin Xie, Xiaoqian Sun, and Huawei Shen. R^2PO: Decoupling training trajectories from inference responses for LLM reasoning. *arXiv preprint arXiv:2601.11960*, 2026a.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

William F Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, Kyunghyun Cho, and Martin Riedmiller. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

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

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From f (x) and g (x) to f (g (x)): Llms learn new skills in rl by composing old ones. In *International Conference on Learning Representations*, volume 2026, pp. 147547–147574, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

A TRAINING & HYPERPARAMETERS

We report all hyperparameters shared by all methods in Table 1. As in the standard DAPO (Yu et al., 2026) algorithm, we perform dynamic sampling, overlong filtering, clip-higher, apply a token-level policy gradient loss, and do not include a KL penalty term.

Table 1: Hyperparameters. Shared by all models and all runs unless noted.

RL (explorer, student, baselines)	
Objective	DAPO clipped objective, ε_{low}=0.2, $\varepsilon_{\text{high}}$=0.28, KL β=0
Prompts per step, group size	B=4, G=16
Steps	explorer 200, student 100, baselines 300, DAPO (4× steps) 1,200 total
Max completion length	32,768
Soft overlong penalty	linear over the final 6,554 tokens
Optimizer	AdamW (β=(0.9, 0.95), no weight decay)
Learning rate (constant)	explorer and DAPO baselines 5×10^{-6} (constant); student 1×10^{-6} (constant)
SFT (student)	
Loss	masked cross-entropy loss (completion-only)
Epochs, max examples	2 epochs, 500 per round
Optimizer	AdamW
Learning rate	5×10^{-6} (constant)
Novelty bonus (RND)	
Layers	$\{\lfloor L/4\rfloor, \lfloor L/2\rfloor, \lfloor 3L/4\rfloor\}$
Predictor and target	3-layer MLP (Linear–ReLU–Linear–ReLU–Linear), width and output 512
Predictor optimizer	Adam, learning rate 10^{-4}
Features	live policy's hidden states, mean-pooled over completion tokens
Bonus	raw (no normalization), credited on verifier-correct completions only

RL update. All RL stages use entirely on-policy steps: vLLM servers reload the trainer's weights after every step and each batch of 4 prompts × 16 samples contributes to the following gradient step.

A.1 COMPUTE BUDGET

Table 2: Update allocation per configuration. Every ExpDis run uses 200 explorer and 100 student updates (19,200 selected rollouts); single-model baselines spend all 300 on one model. Parallel explorers run concurrently, so wall-clock time is equal across rows. DAPO (4× steps) is the only exception.

Configuration	$R \times K$	Explorer updates	Student updates per round
Single-Explorer	1×1	200	100
Breadth K=2	1×2	100 each	100
Breadth K=3	1×3	67/67/66	100
Breadth K=5	1×5	40 each	100
Breadth K=7	1×7	29/29/29/29/28/28/28	100
Depth R=4	4×1	50 per round	25
Depth R=5	5×1	40 per round	20
MR-ME R=4, K=3	4×3	17/17/16 per round	25
MR-ME R=5, K=3	5×3	14/13/13 per round	20
MR-ME R=4, K=5	4×5	10 per explorer per round	25

Hardware. Each run uses one TPU v5litepod-64 slice (16 hosts): 4 hosts train the policy with FSDP and 12 serve rollouts with vLLM in bf16 with a 40,960-token context.

Table 3: Sampling parameters.

	Training rollouts	Evaluation
Temperature	1.0	0.6
top-p / top-k / min-p	0.95 / 20 / –	0.95 / 20 / 0
Max completion tokens	32,768	32,768
Samples per problem	16	64

A.2 PASS@k ESTIMATOR

With n samples per problem, of which c_i are correct on problem i, we use the unbiased estimator of Chen et al. (2021) averaged over N problems:

$$\text{pass@}k \;=\; \frac{1}{N}\sum_{i=1}^{N}\left[1-\frac{\binom{n-c_i}{k}}{\binom{n}{k}}\right], \qquad k \le n. \tag{3}$$

B EXTENDED RESULTS

We report pass@k for all models and methods in Figure B and Table 4

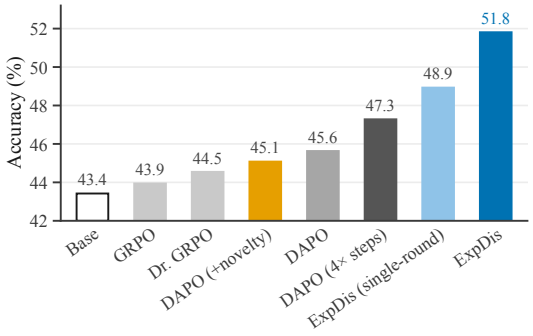


Figure 7: Mean accuracy of each method on Qwen3-1.7B.

Table 4: Full pass@k results.

Model	Method	@1	@2	@4	@8	@16	@32	@64
Qwen3-1.7B	Base	43.42	52.18	58.26	63.02	66.95	70.63	73.82
Qwen3-1.7B	DAPO	45.63	54.31	60.67	65.54	69.16	72.14	74.64
Qwen3-1.7B	DAPO (4× steps)	47.28	56.44	62.83	67.41	70.80	73.62	76.24
Qwen3-1.7B	DAPO (+novelty)	45.08	53.78	60.07	64.91	68.61	71.77	74.43
Qwen3-1.7B	ExpDis (single-round)	48.93	58.55	64.98	69.27	72.43	75.10	77.83
Qwen3-1.7B	ExpDis	51.81	61.40	68.03	72.94	76.97	80.29	83.04
Qwen3-4B	Base	61.06	68.90	72.83	75.43	77.41	78.90	79.86
Qwen3-4B	DAPO	62.58	69.97	73.84	76.46	78.41	79.88	81.14
Qwen3-4B	DAPO (4× steps)	63.64	71.15	75.05	77.65	79.62	81.20	82.82
Qwen3-4B	DAPO (+novelty)	62.20	69.70	73.59	76.20	78.16	79.64	80.82
Qwen3-4B	ExpDis (single-round)	64.69	72.32	76.24	78.83	80.82	82.51	84.50
Qwen3-4B	ExpDis	66.50	74.80	79.13	81.82	83.82	85.53	87.63
Ministral-3-3B	Base	27.95	34.54	40.52	45.34	49.36	53.00	56.50
Ministral-3-3B	DAPO	29.63	36.63	43.21	48.79	53.45	57.19	60.12
Ministral-3-3B	DAPO (4× steps)	30.90	38.19	45.05	50.86	55.74	59.63	62.36
Ministral-3-3B	DAPO (+novelty)	29.21	36.11	42.54	47.93	52.43	56.15	59.22
Ministral-3-3B	ExpDis (single-round)	32.16	39.74	46.87	52.93	58.01	62.05	64.59
Ministral-3-3B	ExpDis	34.76	43.22	51.31	58.35	64.37	69.24	72.56

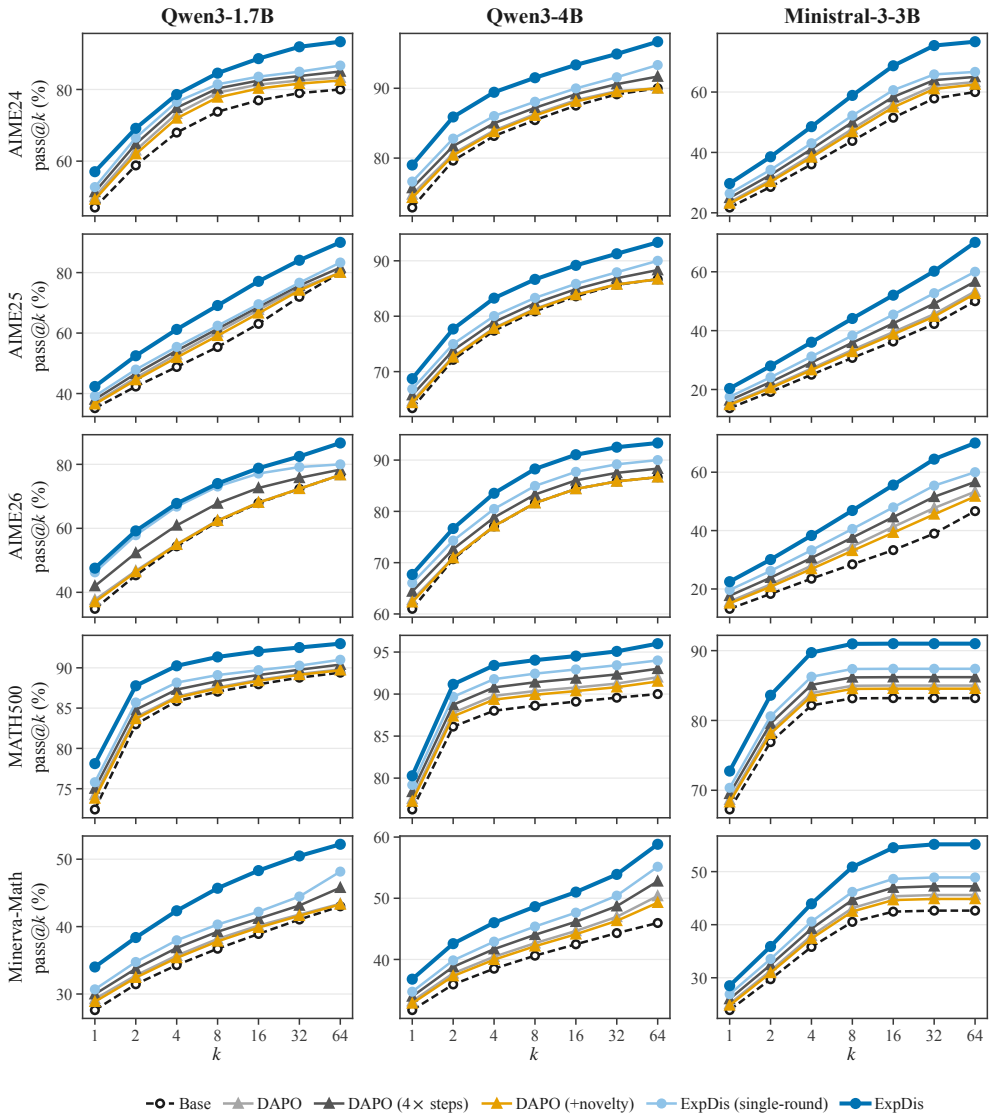


Figure 8: Full pass@k results.

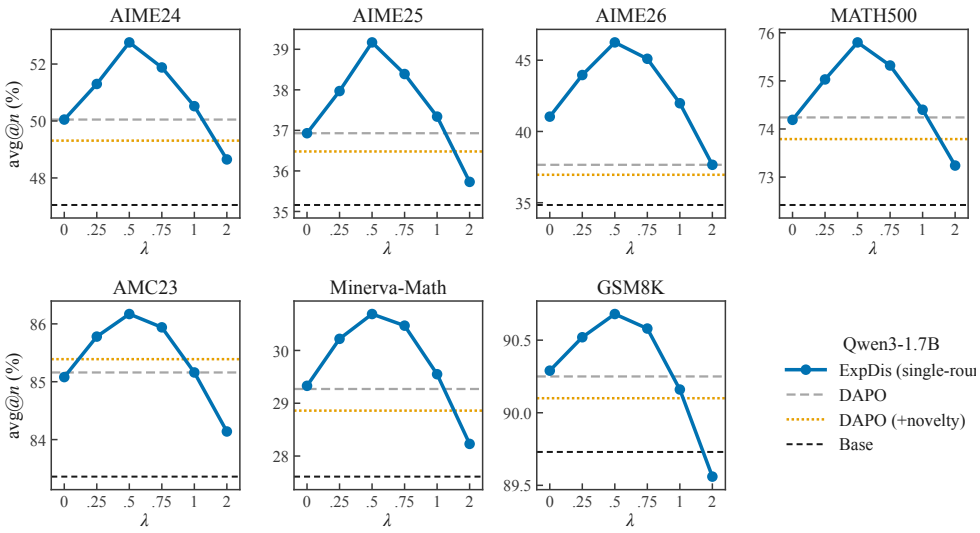


Figure 9: On Qwen3-1.7B, we find that a novelty bonus weight of λ=0.5 on the explorer policy roughly maximizes downstream student policy performance, though several values of λ outperform the DAPO baseline.

B.1 Novelty metric (r_{novelty})

To test whether our gains depend on RND, we train two single-round explorers on Qwen3-1.7B with a different novelty term based on the mean-pooled, L2-normalized final-layer hidden states of each trajectory.

1. *kNN novelty* scores each correct completion by its mean cosine distance to its k=16 nearest neighbors in a replay buffer of the 4,096 most recent correct completions.
2. *Elliptical novelty* (Tuyls et al., 2026) uses the same features ϕ and the bonus $\sqrt{\phi^\top \Sigma^{-1} \phi}$, where Σ is a ridge-regularized running covariance.

We use λ=0.5 for both and report results in Table 5.

Table 5: We ablate the choice of novelty metric used as our novelty bonus.

Method	r_{novelty}	AIME24	AIME25	AIME26	MATH500	Minerva	Mean
DAPO	–	50.05	36.92	37.66	74.24	29.27	45.63
DAPO (4× steps)	–	51.41	38.05	41.96	75.02	29.98	47.28
ExpDis (ours)	RND	52.76	39.17	46.25	75.80	30.69	48.93
ExpDis	kNN	50.83	37.50	43.75	75.00	29.75	47.37
ExpDis	Elliptical	51.56	38.23	44.79	75.30	30.40	48.06

C Scaling Analysis

D Diversity Analysis

For completeness we restate the three diversity metrics used in the paper. In tables they appear as Inter4 or Ans@n, prefixed Explorer- or Final- by stage.

InterDistinct-4. Given n generations $\{g_1, \ldots, g_n\}$ for a single problem, let $4\text{grams}(g_i)$ be the multiset of 4-grams in g_i and $G_4 = \bigcup_i 4\text{grams}(g_i)$ the set of distinct 4-grams in the pool. Then

$$\text{InterDistinct-4} = \frac{|G_4|}{\sum_i |4\text{grams}(g_i)|},$$

the ratio of distinct 4-grams to total 4-grams in the pool. We average across problems.

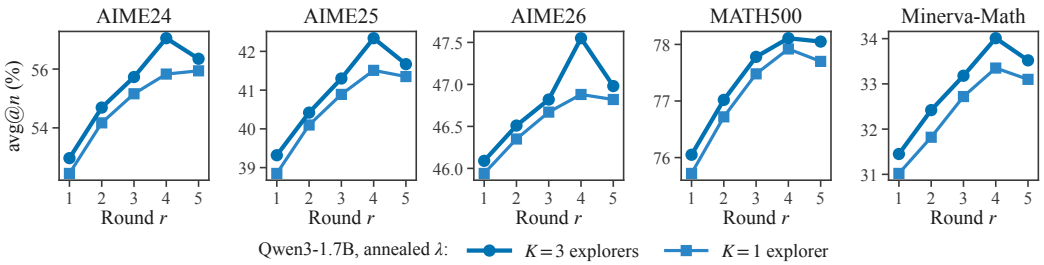


Figure 10: Breakdown of impact of increasing rounds given a fixed number of explorers per round.

AnswerDistinct@n. For each problem, count the number of distinct extracted final answers across the n generations, then average across problems.

Token entropy. For each model we sample 8 rollouts for each of the 30 AIME24 problems with the evaluation sampler and record the mean per-token entropy of the sampled tokens. Table 6 shows ± 1 standard error from a bootstrap over problems.

Strategy diversity. We follow (Li et al., 2025), who finetune Qwen3-Embedding-4B to maximize the similarity between mathematical reasoning traces judged as being semantically similar in approach by a strong LLM. We use their checkpoint to produce pairwise similarity scores between all generations for a single prompt. We then cluster the generations and report the resulting number of clusters C normalized by the number of generations.

Table 6: We report several diversity metrics across several Qwen3-1.7B models on generations on AIME24.

Model	Entropy (nats)	Semantic	Ans. entropy	Distinct ans.	InterDistinct-4
Base	0.237 ± 0.010	–	–	–	–
DAPO	0.250 ± 0.011	0.940	1.30	12.0	–
DAPO ($4\times$ steps)	0.266 ± 0.012	0.942	1.373	12.10	0.247
ExpDis (single-explorer)	0.307 ± 0.012	0.960	1.138	10.47	0.236
ExpDis	0.455 ± 0.061	0.978	1.499	15.47	0.221

Table 7: We report several diversity metrics on the student policy on AIME24 generations for different λ values on the explorer policy.

		Explorer		Student	
Model	Method	Inter4	Ans@n	Inter4	Ans@n
Qwen3-1.7B	Base	–	–	0.311	8.94
Qwen3-1.7B	DAPO	–	–	0.308	8.50
Qwen3-1.7B	DAPO (+novelty, λ=0.5)	–	–	0.342	9.39
Qwen3-1.7B	ExpDis (λ=0.5)	0.422	12.61	0.357	9.93
Qwen3-1.7B	ExpDis (λ=0.75)	0.478	14.44	0.363	10.02
Qwen3-1.7B	ExpDis (λ=1.0)	0.534	16.28	0.335	9.30
Qwen3-1.7B	ExpDis (λ=2.0)	0.550	23.61	0.283	7.87
Qwen3-4B	Base	–	–	0.290	4.02
Qwen3-4B	DAPO	–	–	0.287	3.82
Qwen3-4B	DAPO (+novelty, λ=0.5)	–	–	0.319	4.22
Qwen3-4B	ExpDis (λ=0.5)	0.394	5.67	0.333	4.47

E Ablations

Quality filtering. With three explorers, keeping every correct trajectory from the pooled explorers (NaivePool) gives a mean of 48.95%, no better than a single explorer (48.93%). Applying the quality filter increases performance to 49.47%.

Student distillation vs student RL. On Qwen3-1.7B, distilling unfiltered explorer traces and then running RL reaches 49.79% on AIME24, and distilling filtered traces without RL reaches 49.84%. Both distillation and RL taken alone underperform DAPO (at 50.05%). Only filtering followed by RL reaches 52.76% (Table 8).

Annealing λ. We find that annealing λ during multiple rounds $(0.75, 0.50, 0.35, 0.25)$ outperforms a fixed λ=0.5 at every round (see Figure 11).

Table 8: Ablating student distillation, student RL, or both.

Configuration	AIME24
DAPO	50.05
Unfiltered SFT + RL	49.79
Filtered SFT, no RL	49.84
Filtered SFT + RL (ExpDis)	52.76

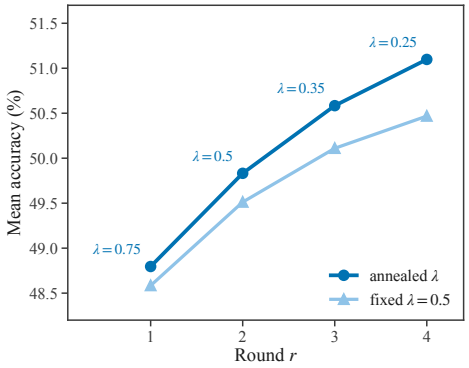


Figure 11: Fixed versus annealed λ across four rounds (Qwen3-1.7B, single-explorer).

Training data. Every run samples the same number of prompts (about 1,200), so these runs differ only in the pool they sample from (Figure 12, Table 9). Sampling from DAPO-Math-17K or a 17K subset of DeepScaleR gives similar results, with neither better on every benchmark. Sampling ExpDis prompts from the full 40K DeepScaleR pool is better on every benchmark. All main results use DAPO-Math-17K.

Table 9: We ablate the choice of training dataset. We see roughly the same performance across all methods.

Method	Dataset	AIME24	AIME25	AIME26	MATH500	AMC23	Minerva	GSM8K
GRPO	DAPO-Math-17K	47.81	35.36	35.68	72.84	83.75	28.02	89.89
GRPO	DeepScaleR-17K	47.96	35.52	35.68	72.71	84.09	27.86	89.81
Dr. GRPO	DAPO-Math-17K	48.49	35.99	36.46	73.34	84.38	28.45	90.04
Dr. GRPO	DeepScaleR-17K	48.83	36.26	36.51	73.18	84.78	28.20	89.95
DAPO	DAPO-Math-17K	50.05	36.93	37.66	74.24	85.16	29.27	90.25
DAPO	DeepScaleR-17K	49.68	37.21	37.76	74.08	85.62	28.90	90.14
ExpDis λ=0.5	DAPO-Math-17K	52.76	39.17	46.25	75.80	86.17	30.69	90.68
ExpDis λ=0.5	DeepScaleR-17K	51.77	38.33	44.84	75.16	86.48	30.44	90.59
ExpDis λ=0.5	DeepScaleR-full	52.89	39.24	47.66	76.08	86.84	31.37	90.78

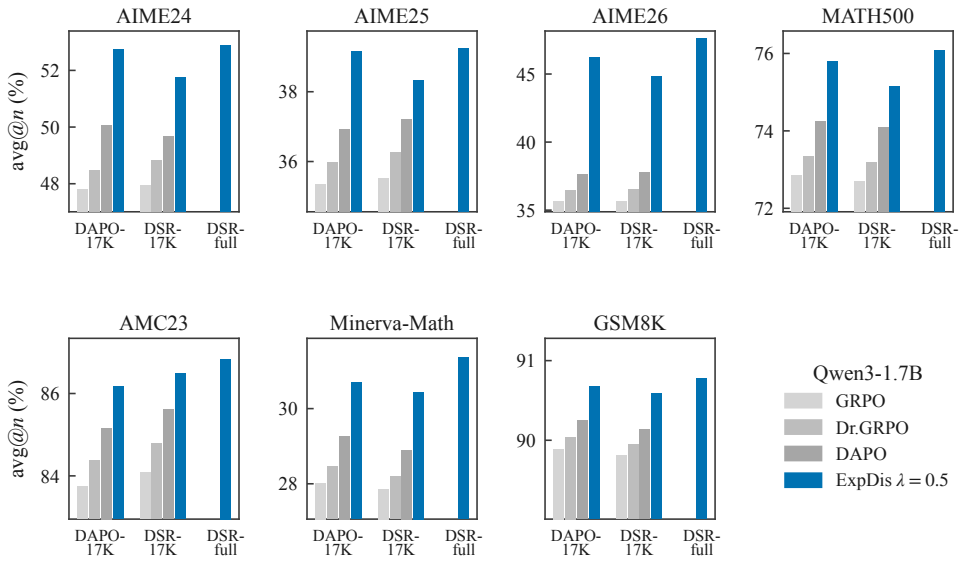


Figure 12: We ablate the choice of training dataset. We see roughly the same performance across all methods.