Towards Effective Experiential Learning: Dual Guidance for Utilization and Internalization

Anonymous ACL submission

Abstract

Reinforcement learning from verifiable rewards (RLVR) has become a promising paradigm for improving the reasoning capabilities of large language models (LLMs). However, existing RL-based training remains only a rough approximation to human learning, where humans leverage external experience to guide exploration and gradually internalize problem-solving strategies into stable knowledge. Motivated by this gap, we ask: how can LLMs better utilize and internalize experience during RLVR training? To answer this question, we propose **D**ual **G**uidance **O**ptimization (**DGO**), which treats experience as a bridge between external guidance and internal policy learning. Rather than using experience only as an inference-time aid or trajectories only as optimization signals, DGO couples them in a closed loop: experience guides exploration, improved trajectories renew the experience bank, and guided trajectories are rewritten as self-contained traces for policy distillation. This enables the model to better utilize external experience while yielding consistent gains under experience-free inference. Experiments across multiple benchmarks show that DGO consistently outperforms baselines under both experience-free and experience-guided inference, suggesting that coupling utilization and internalization leads to effective reasoning.

1 Introduction

Recently, reinforcement learning (RL) has shown great promise in enhancing large language models (LLMs) (Zhao et al., 2023; Guo et al., 2025), especially through reinforcement learning from verifiable rewards (RLVR). By optimizing LLMs with outcome-based rewards, RLVR encourages correct trajectories and suppresses incorrect ones, thereby improving reasoning capabilities (Shao et al., 2024; Deng et al., 2025). However, standard RLVR mainly treats sampled trajectories as optimization signals, leaving the policy to discover useful strategies largely on its own. When the policy lacks effective strategies or diagnostic feedback, exploration can become inefficient and repeatedly produce similar failures. This motivates organizing historical trajectories into reusable experience for guiding future exploration, rather than using them only for parameter updates.

Experiential learning provides a natural perspective for this problem. Inspired by human learning, experience can be distilled from prior problem-solving attempts or acquired from external sources, such as books, examples, or domain experts (Kolb, 2014). It can then serve two complementary roles: guiding future exploration as an external reference and being gradually internalized into stable knowledge (Shirouzu et al., 2002). Recent studies have incorporated experience into LLM training (Zhan et al., 2025; Tang et al., 2025; Wang et al., 2025), showing its potential for improving policy exploration and capability acquisition.

Nevertheless, existing methods often capture only part of this experiential learning process. Parametric methods, including supervised fine-tuning and RL-based training (Ouyang et al., 2022; Yu et al., 2025; Xi et al., 2025), internalize useful behaviors into model parameters, but usually treat trajectories as optimization signals rather than reusable experience. In contrast, external-guidance methods provide decomposition, hints, memory retrieval, or accumulated experience during reasoning (Zhao et al., 2024; Cai et al., 2025; Xu et al., 2025). While such guidance can ease the current rollout, it may leave the model dependent on explicit external information. Therefore, a more complete experiential RLVR process should couple *experience utilization* and *experience internalization*, enabling the model to use external experience while absorbing induced reasoning patterns for future experience-free inference (Gao et al., 2025).

To address this, we propose **D**ual **G**uidance

Optimization (**DGO**), which treats experience as a bridge between external guidance and internal policy learning. DGO couples experience utilization and internalization through a closed loop. It first abstracts historical trajectories into reusable experience, enabling the policy to explore beyond what it can discover on its own. The resulting improved trajectories are then used to renew the experience bank, providing more informative guidance for subsequent exploration. Finally, experience-guided trajectories are rewritten into self-contained traces and distilled into the policy, aiming to transfer experience-induced reasoning patterns into experience-free inference. Through this iterative process, experience utilization and internalization mutually reinforce each other, leading to more effective reasoning.

The contributions are summarized as follows:

- We show that coupling experience-guided exploration with distillation improves both experience-guided and experience-free reasoning, suggesting that experience supports inference-time guidance and policy learning.
- We introduce Dual Guidance Optimization (DGO), a closed-loop training method that formalizes the interaction between non-parametric experience and parametric training, offering a unified view of experience utilization and internalization.
- DGO achieves the best average score of 32.41% across six challenging benchmarks on Qwen3-8B-Base under intrinsic inference and 39.38% under test-time scaling, demonstrating its effectiveness in both experience-free and experience-guided reasoning.

2 Related Work

External Experience Guidance. External experience guidance improves LLM reasoning by supplying non-parametric information during generation, including historical memories (Chhikara et al., 2025; Salama et al., 2025), expert demonstrations (Brown et al., 2020), and retrievable experience (Cai et al., 2025; Tang et al., 2025; Yang et al., 2025b). Recent work further refines experience through updating and verification mechanisms (Cao et al., 2025; Kirtania et al., 2025), or studies how agents exploit such experience during reasoning (Xia et al., 2026; Zhang et al., 2026; Shi et al., 2026). While these methods ease rollout with external guidance, they mainly treat experience as an inference-time resource. DGO instead uses experience to guide exploration and internalizes the resulting trajectories through training.

Parametric Experience Internalization. Parametric experience internalization improves LLM reasoning by updating model parameters to reinforce useful behaviors. Representative methods include supervised fine-tuning, reinforcement learning (Ouyang et al., 2022; Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025; Zhan et al., 2025), preference optimization (Rafailov et al., 2023), and verifier-based training (Lightman et al., 2023). Self-improvement and distillation methods (Zelikman et al., 2022; Gulcehre et al., 2023; Didolkar et al., 2025; Zhao et al., 2026; Huang et al., 2026; Lu et al., 2026) further train on model-generated rationales or selected samples to internalize useful reasoning behaviors. However, these methods usually treat trajectories as optimization targets rather than reusable experience for later exploration. DGO bridges this gap by coupling trajectory-based internalization with a renewable experience bank, so that trajectories not only update the policy but also improve future experience-guided exploration.

3 Method

In this section, we present the details of DGO.

3.1 Overview of Dual Guidance Optimization

Dual Guidance Optimization (DGO) is based on a simple observation: reasoning can be improved by two complementary sources of guidance. External guidance helps the model explore solution trajectories that are difficult to discover under the current policy alone, while internal guidance, encoded in the model parameters, supports the model's own reasoning capability. DGO links the two by transforming trajectories discovered under external guidance into parameter updates, encouraging useful reasoning patterns to be transferred into the policy.

Formally, given an input problem x, an experience bank $\mathcal{E}$, and a policy π_θ, the model generates trajectories under the joint effect of external and internal guidance as

$$s \sim \pi_\theta(\cdot \mid x, \mathcal{E}). \tag{1}$$

The resulting trajectories are then transformed into supervision for parameter update:

$$\theta' \leftarrow \mathrm{Update}(\theta; \mathcal{T}(s)), \tag{2}$$

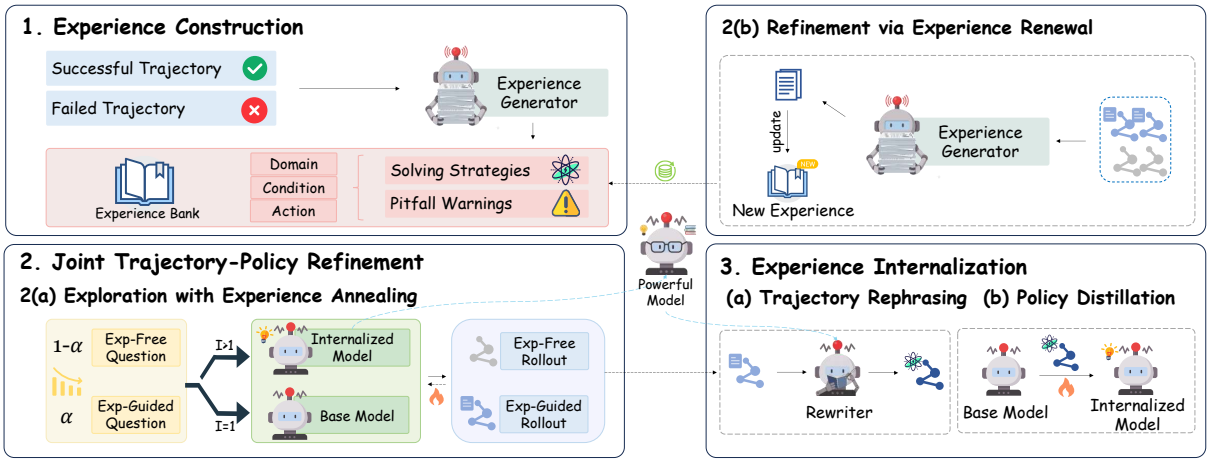


Figure 1: Our framework consists of three components: 1) Experience Construction, 2) Joint Trajectory–Policy Refinement, and 3) Experience Internalization, forming a closed-loop process that enhances experience utilization and internalization. I denotes the training iteration, and the final model is the RL policy from the last iteration.

where $\mathcal{T}(\cdot)$ denotes the transformation from generated trajectories to training signals, such as filtering and rewriting. Through this process, external and internal guidance form a closed loop in which external experience improves trajectory discovery, and the resulting trajectories in turn refine experience, strengthen the model's internal guidance, and support effective use of external experience.

3.2 Iterative Dual-Guidance Training

Building on Section 3.1, we implement an iterative training cycle, illustrated in Figure 1, consisting of experience construction, joint trajectory-policy refinement, and experience internalization. The pseudocode is shown in Appendix A.

3.2.1 Experience Construction

Experience is distilled from historical reasoning trajectories to provide external guidance for future problem solving. Rather than directly reusing full reasoning trajectories, we extract abstract condition-action guidance that captures transferable strategies and diagnostic cues. Although derived from individual trajectories, such guidance is represented as reusable reasoning patterns rather than problem-specific solution traces. Formally, each experience is represented as a triplet $e = (c, \kappa, \tau)$, where c specifies the broader domain or problem phase, κ identifies the key condition, scenario, or potential error, and τ prescribes the corresponding action to take or avoid.

Given a trajectory s, a unified experience generator G extracts an experience as $e = G(s)$. Experiences derived from correct trajectories capture effective solving strategies, while those derived from incorrect trajectories capture pitfall warnings. Collectively, these experiences form the bank $\mathcal{E} = \{e_i\}_{i=1}^{N}$, which aggregates guidance from many trajectories and provides a bank-level source of external guidance. To ensure that the bank provides transferable reasoning guidance rather than answer-specific supervision, any experience that contains or strongly suggests the final answer is strictly filtered out. The extraction prompt is shown in Table 12, and examples are provided in Table 16.

3.2.2 Joint Trajectory-Policy Refinement

In this stage, we employ RL (*e.g.,* GRPO) to strengthen experience-guided reasoning while continuously producing higher-quality trajectories for subsequent experience internalization.

Exploration with Experience Annealing. We use RL to strengthen the model's ability to utilize external experience while gradually reducing its dependence on such guidance. To this end, at iteration k, training draws instances from a mixture of the experience-guided distribution $D_e^{(k)}$ and the experience-free distribution $D_i^{(k)}$, where the annealing coefficient α_k controls the strength of experience guidance:

$$D^{(k)} = \alpha_k D_e^{(k)} + (1 - \alpha_k) D_i^{(k)}. \qquad (3)$$

As training progresses, α_k gradually decreases, so that the policy relies more heavily on external experience in earlier stages for guided exploration, and progressively shifts toward weaker guidance in later stages. Let $z = (x, \tilde{e})$ denote a training

instance, where $\tilde{e}$ is the retrieved experience for experience-guided instances and is empty otherwise. The policy is optimized over the mixed training distribution using

$$\mathcal{L}(\theta) = \mathbb{E}_{z\sim D^{(k)},\, s\sim\pi_{\theta_{\text{old}}}(\cdot|z)}\left[\frac{1}{|s|}\sum_{t=1}^{|s|}\min\Big(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t\Big)\right], \tag{4}$$

where $\rho_t(\theta) = \frac{\pi_\theta(s_t|z,s_{<t})}{\pi_{\theta_{\text{old}}}(s_t|z,s_{<t})}$. This annealing process encourages the model to benefit from strong experience guidance early on, while gradually reducing reliance on external experience without losing the ability to use it effectively.

Refinement via Experience Renewal. After each RL iteration k, a set of rollout trajectories $\mathcal{S}^{(k)} = \{s_j^{(k)}\}$ is collected. The experience generator G summarizes these trajectories into a new batch of experience: $\mathcal{E}_{\text{new}}^{(k)} = G\big(\mathcal{S}^{(k)}\big)$, which is merged with the current bank and filtered through quality-based selection as shown in Appendix C.1:

$$\mathcal{E}^{(k+1)} = \text{Select}\Big(\mathcal{E}^{(k)} \cup \mathcal{E}_{\text{new}}^{(k)}\Big), \tag{5}$$

where $\text{Select}(\cdot)$ retains the most informative experiences for the next iteration. The updated bank $\mathcal{E}^{(k+1)}$ is then used to construct the mixed training distribution $D^{(k+1)}$ for later policy refinement.

As trajectory quality improves across iterations, the extracted experience becomes increasingly useful for guiding later exploration. This creates a positive feedback loop: improved trajectories yield higher-quality experience, which in turn supports further trajectory refinement and provides better data for subsequent experience internalization.

3.2.3 Experience Internalization

Experience internalization aims to transfer reasoning gains induced by external experience into the policy parameters through *trajectory rephrasing* and *policy distillation*. In this way, the model is encouraged to preserve useful reasoning patterns from experience-guided trajectories, improving its reasoning ability without explicit experience.

Trajectory Rephrasing. External experience can guide the model to produce trajectories that would be difficult to obtain through unguided reasoning alone. However, such trajectories often contain explicit references to the provided experience, which can introduce undesirable dependencies if used directly for training. To address this, we rewrite experience-guided trajectories into self-contained reasoning traces that remove explicit references while preserving their underlying rationale.

Given a problem x with retrieved experience $\mathcal{E}_x$ and an experience-guided trajectory $s_e^{(k)} \in \mathcal{S}_e^{(k)}$, we use the RL checkpoint $\theta_{\text{RL}}^{(k)}$ with a rewriting prompt template p (see in Table 15) to generate a rewritten trajectory:

$$\tilde{s}_e^{(k)} \sim \pi_{\theta_{\text{RL}}^{(k)}}\Big(\cdot \mid x, s_e^{(k)}, \mathcal{E}_x, p\Big). \tag{6}$$

Applying this procedure to all trajectories in $\mathcal{S}_e^{(k)}$ yields the rewritten trajectory set $\tilde{\mathcal{S}}_e^{(k)}$. A rewritten trajectory is retained only if it satisfies strict filtering constraints, including correctness validation, keyword exclusion, and length control, with detailed settings provided in Appendix C.2. Denoting the filtering operator by $\mathcal{F}(\cdot)$, the implicit trajectory set at iteration k is defined as

$$\mathcal{S}_{\text{imp}}^{(k)} = \text{Top-K}\Big(\mathcal{F}\big(\tilde{\mathcal{S}}_e^{(k)}\big)\Big), \tag{7}$$

where up to K solutions are retained per problem.

Policy Distillation. We further internalize external experience into the model parameters through supervised distillation over selected trajectories. At iteration k, we accumulate both implicit and experience-guided trajectories as $\mathcal{S}_{\text{imp}}^{(\le k)} = \bigcup_{i=1}^{k}\mathcal{S}_{\text{imp}}^{(i)}$ and $\mathcal{S}_e^{(\le k)} = \bigcup_{i=1}^{k}\mathcal{S}_e^{(i)}$, respectively. To preserve the model's ability to utilize external experience, we construct the distillation dataset by mixing these two trajectory sets:

$$\mathcal{D}_{\text{dist}}^{(k)} = \beta_k\,\mathcal{S}_{\text{imp}}^{(\le k)} + (1-\beta_k)\,\mathcal{S}_e^{(\le k)}, \tag{8}$$

where $\beta_k \in [0,1]$ controls the mixing ratio. The policy is then optimized by maximum likelihood, with parameters initialized from the initial checkpoint $\theta^{(0)}$ to mitigate overfitting:

$$\theta^{(k+1)} = \arg\max_{\theta}\ \mathbb{E}_{(x,s)\sim\mathcal{D}_{\text{dist}}^{(k)}}\ \log\pi_\theta(s \mid x). \tag{9}$$

The resulting model, denoted as the *Internalized Model*, is then used to initialize the next iteration.

3.3 Inference Paradigm

Intrinsic Inference. This mode evaluates the model's reasoning ability without external guidance. Given a problem x, the model generates a trajectory solely from its parametric knowledge:

$$s \sim \pi_\theta(\cdot \mid x). \tag{10}$$

Iterative Experience-Guided Test-Time Scaling. Given a problem x, the model performs K rounds of trajectory generation. In each round k, it produces N trajectories, from which the experience generator G extracts experience $\mathcal{E}_{\text{online}}^{(k)}$ for the next round. The first round is performed without external experience, while later rounds use the online experience extracted from the previous round[1]:

$$s^{(k)} \sim \begin{cases} \pi_\theta(\cdot \mid x), & k = 1, \\ \pi_\theta(\cdot \mid x, \mathcal{E}_{\text{online}}^{(k-1)}), & k > 1. \end{cases} \tag{11}$$

4 Experiments

In this section, we first detail the experimental setup and then report the results and detailed analysis.

4.1 Experiment Settings

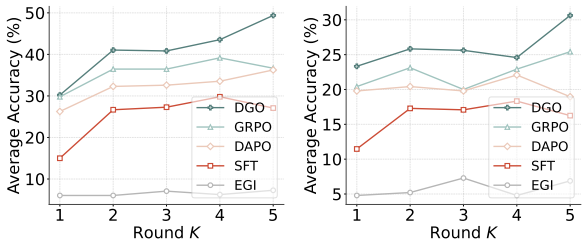
Training. We use Qwen3 models (Yang et al., 2025a) at three scales (4B, 8B, and 14B) as policy backbones, and sample 9,600 questions from SkyWork (He et al., 2025) for RL training. Training consists of three GRPO-based iterations with iterative experience update and ratio annealing: the experience ratio α decreases from 1.0 to 0.5 and 0.25, while the internalization ratio β_k is fixed at 0.5. Appendix C.3.3 analyzes α and β, showing that this setting yields stable performance and a favorable balance. After each iteration, selected trajectories are rewritten and distilled into the policy. The experience generator is obtained by finetuning Qwen3-8B-Base on 40,000 samples distilled from DeepSeek v3.2 (Liu et al., 2025), and is used throughout training; see Appendix B for details. Appendix C.3.2 further shows that DGO uses an RL budget comparable to GRPO and incurs only a 1.15× total training-cost overhead, suggesting a favorable cost-performance trade-off.

Evaluation. We evaluate our method on in-domain benchmarks, including AIME24, AIME25, HMMT25-Feb, and MATH500 (Hendrycks et al., 2021), and out-of-domain benchmarks, including GPQA-Diamond (Rein et al., 2024) and HLE-Verified (Zhai et al., 2026). Unless otherwise specified, we use temperature 1.0, top-p 1.0, a maximum sequence length of 16K, and report avg@16. We consider two settings: intrinsic inference, which measures standalone reasoning, and Test-Time Scaling (TTS) (Hu et al., 2026), which evaluates experience-guided inference. For TTS, we use at most $K = 5$ iterations, summarize up to $N = 4$ experiences per iteration.

Baselines. We compare DGO with both training-based and training-free baselines. For training-based baselines, we include SFT, GRPO (Shao et al., 2024), and DAPO (Yu et al., 2025), which optimize model parameters without using external experience during training. For training-free baselines, we consider Experience-Guided Inference (EGI), which conditions the model on external experience without parameter updates. Beyond EGI, we apply the same TTS protocol to models from different training methods to compare their ability to utilize external experience.

4.2 Main Results

DGO consistently enhances intrinsic reasoning across model scales. As shown in Table 1, DGO (zero) consistently improves over baselines, including SFT and RL-based methods, achieving the best average scores of 30.34% for 4B, 32.41% for 8B, and 38.21% for 14B, with improvements on both challenging mathematical benchmarks, including AIME25 and HMMT25, and broader-coverage benchmarks like MATH500. These results provide evidence that experience-guided exploration and distillation improve experience-free reasoning, leading to consistent intrinsic gains.



(a) AIME24 (b) AIME25

Figure 2: Accuracy of DGO and baseline methods across successive TTS rounds on AIME24 and AIME25 using Qwen3-8B-Base.

DGO enables more effective use of experience. DGO also demonstrates strong experience-guided reasoning under TTS. As shown in Figure 2, performance improves steadily across successive TTS rounds and consistently surpasses other trained policies. Table 7 further compares different policies under the same TTS protocol, showing that DGO more effectively utilizes external experience during

[1]The experience generator does not access correctness labels and must infer correctness before summarizing experience; see Table 12.

Model	Method	In-domain (%)				Out-of-domain (%)		Avg
		AIME24	AIME25	HMMT25	MATH500	GPQA-D	HLE	
Qwen3-4B-Base	EGI	1.46	1.46	0.00	23.52	12.12	1.31	6.64
	SFT	5.83	5.62	1.04	31.52	22.92	2.56	11.58
	GRPO	26.88	22.50	8.75	**72.75**	42.55	2.73	29.36
	DAPO	26.04	22.29	9.17	71.35	41.79	2.34	28.83
	DGO (zero)	**27.29**	**23.54**	**11.67**	72.60	**43.56**	**3.37**	**30.34**
	DGO (TTS)	40.21	27.71	15.83	78.35	52.53	4.45	36.51
Qwen3-8B-Base	EGI	6.04	4.79	2.08	41.50	22.22	3.12	13.29
	SFT	15.00	11.46	3.54	70.38	36.30	4.34	23.50
	GRPO	29.79	20.42	9.17	73.20	44.32	3.95	30.14
	DAPO	25.00	19.17	9.17	**80.77**	45.83	**4.66**	30.77
	DGO (zero)	**30.21**	**23.33**	**11.88**	75.25	**49.75**	4.02	**32.41**
	DGO (TTS)	49.38	30.63	14.37	80.42	54.99	6.51	39.38
Qwen3-14B-Base	EGI	8.33	5.42	1.67	47.02	28.91	3.29	15.77
	SFT	18.12	13.33	6.46	74.58	37.12	4.96	25.76
	GRPO	38.12	27.08	14.37	89.00	45.01	5.15	36.45
	DAPO	35.83	26.25	14.17	89.50	49.37	5.45	36.76
	DGO (zero)	**38.54**	**29.17**	**15.21**	**90.60**	**50.13**	**5.63**	**38.21**
	DGO (TTS)	54.48	39.38	22.29	93.17	51.64	7.71	44.78

Table 1: Performance comparison of our method with other approaches across three model scales on both in-domain and out-of-domain benchmarks. **Bold** indicates the best result within each model, and underlined indicates the second-best. DGO (zero) and DGO (TTS) denote intrinsic inference and test-time scaling, respectively.

inference. Under this setting, DGO reaches average scores of 36.51% on the 4B model and 44.78% on the 14B model. A more detailed analysis of experience utilization is provided in Section 5.1.

DGO shows out-of-domain reasoning transfer. Although trained exclusively on mathematical data, DGO achieves competitive performance on knowledge-intensive benchmarks, including GPQA-Diamond and HLE-Verified. Both DGO (zero) and DGO (TTS) perform well on these tasks, suggesting two complementary effects. First, experience learned from mathematical reasoning may support broader reasoning tasks. Second, the ability to leverage prior reasoning experience may also transfer beyond the training domain.

4.3 Ablation Studies

In this section, we conduct ablation studies to evaluate the contribution of key design choices in DGO.

4.3.1 Component Ablation

We conduct component-level ablations under the same training stage, RL budget, and data scale, with each variant removing only one component for fairness. Intrinsic inference and experience-guided TTS evaluate experience internalization and utilization, respectively. As shown in Table 2, removing any component hurts performance, providing controlled evidence for its contribution.

Method	AIME25	HMMT25	MATH500	GPQA-D	Avg
DGO (zero)	**21.04**	**9.17**	**70.08**	**40.47**	**35.19**
w/o *EA*	17.50	7.71	68.08	38.64	32.98
w/o *ER*	18.75	7.50	68.53	38.57	33.34
w/o *TR*	17.50	6.04	66.65	39.90	32.52
w/o *PD*	20.83	7.71	67.73	38.26	33.63
DGO (TTS)	**26.67**	**15.42**	**73.90**	**52.65**	**42.16**
w/o *EA*	24.58	13.12	70.43	48.30	39.11
w/o *ER*	23.75	15.21	70.78	48.42	39.54
w/o *TR*	20.42	11.67	72.10	48.11	38.08
w/o *PD*	22.71	15.00	67.00	49.31	38.51

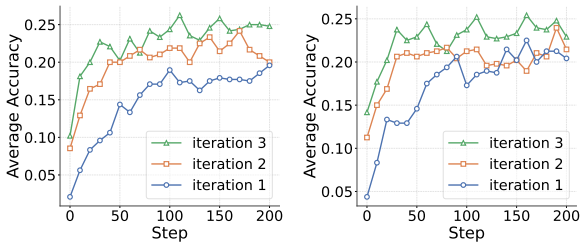
Table 2: Component ablation study on Qwen3-4B-Base. *EA*, *ER*, *TR*, and *PD* denote *Experience Annealing*, *Experience Renewal*, *Trajectory Rephrasing*, and *Policy Distillation*. Experiments are conducted between iteration 1 and 2. Full results are shown in Table 9.

1) *Experience Renewal* keeps the experience pool informative and diverse throughout training. Removing it lowers the average score from 35.19% to 33.34% for DGO (zero) and from 42.16% to 39.54% for DGO (TTS), showing that maintaining fresh experience supports stronger subsequent trajectories and better experience-free improvement.

2) *Trajectory Rephrasing* improves training trajectories by removing noisy patterns that explicitly copy experience, making them easier to internalize.

Without it, the average score drops from 35.19% to 32.52% for DGO (zero) and from 42.16% to 38.08% for DGO (TTS), suggesting that effective trajectory rephrasing can be achieved by the model itself, without requiring a stronger external rewriter.

3) *Experience Annealing* and *Policy Distillation* are important for reducing reliance on explicit experience and improving experience-free policy behavior. Annealing encourages the model to rely less on explicit experience, while distillation consolidates experience-guided trajectories into the policy. Removing annealing drops the average score from 35.19% to 32.98% for DGO (zero) and from 42.16% to 39.11% for DGO (TTS), and removing distillation also causes clear degradation.



(a) Qwen3-8B-Base (b) Qwen3-4B-Base

Figure 3: Accuracy curves of two models on AIME25 across training iterations.

4.3.2 Role of Iterative Training

We further examine the role of iterative training itself. As shown in Figure 3, from iteration 1 to 3, each successive round starts from a higher initial accuracy and maintains a stronger performance curve over most training steps. This suggests that experience accumulated in earlier iterations provides increasingly useful guidance for later training, highlighting the importance of iterative training in DGO. Appendix E.4 provides a case study showing how experience quality improves across iterations.

4.3.3 Cross-Generator Sensitivity Analysis

To examine whether DGO's gains mainly come from the trained experience generator, we conduct a cross-generator sensitivity analysis with three choices: DeepSeek-v3.2-distilled, Qwen3-4B-Instruct, and the policy itself as a self-generator.

As shown in Table 3, the self-generator already improves the average score from 24.64 to 26.51 over GRPO under intrinsic inference. Replacing the distilled generator with 4B-Ins yields comparable intrinsic performance, with the average score changing only slightly from 28.32 to 28.04, and

Method	Generator	AIME25	HMMT25	GPQA-D	Avg
Training-time generator sensitivity under intrinsic inference					
GRPO	-	20.42	9.17	44.32	24.64
DGO	Self	21.73	10.76	47.05	26.51
DGO	Distilled	**23.33**	**11.88**	49.75	**28.32**
DGO	4B-Ins	23.12	10.94	**50.07**	28.04
Test-time generator sensitivity under the same TTS protocol					
GRPO	Self	22.92	10.83	47.79	27.18
DGO	Self	30.00	12.08	52.44	31.51
GRPO	Distilled	25.42	14.17	52.97	30.85
DGO	Distilled	**30.64**	14.37	54.99	**33.33**
GRPO	4B-Ins	22.29	12.92	52.71	29.31
DGO	4B-Ins	28.33	**15.00**	**56.42**	33.25

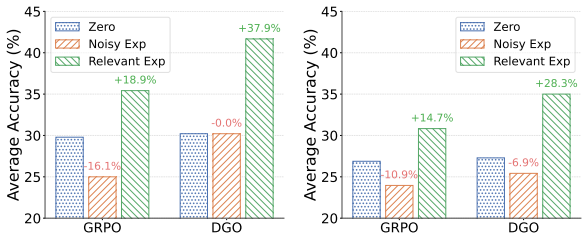
Table 3: Cross-generator sensitivity analysis on Qwen3-8B-Base. The upper block varies the generator used during training and evaluates intrinsic inference; the lower varies the generator used for TTS under the same protocol. *Self* denotes using the policy itself as the generator, *Distilled* denotes the DeepSeek-v3.2-distilled, and *4B-Ins* denotes Qwen3-4B-Instruct.

both outperforming GRPO. Under TTS, DGO consistently outperforms GRPO with the same generator and protocol across all three generator choices. Notably, Distilled and 4B-Ins yield comparable DGO performance, while Self also provides clear gains over its GRPO counterpart. These results indicate that DGO does not rely on a particular generator; instead, the improvements are consistently supported by the learning loop across both intrinsic and experience-guided settings.

5 Analysis

In this section, we analyze the role of experience in our method from two complementary aspects: *utilization* and *internalization*. We also provide case studies in Appendix E.2 and Appendix E.3 to offer intuitive evidence for both capabilities.

5.1 The Role of Experience in Utilization



(a) Qwen3-8B-Base (b) Qwen3-4B-Base

Figure 4: Experience utilization under three settings on AIME24. DGO benefits more from relevant experience and is more robust to irrelevant experience. Numbers denote relative improvement over zero experience.

Even high-quality experience cannot be fully exploited if the model cannot effectively utilize it (Dou et al., 2026). We compare DGO- and GRPO-trained models under three settings: *without experience*, *with noisy experience*, and *with relevant experience*. Relevant experience is summarized from historical trajectories on the same problem, while noisy experience is from those of irrelevant problems. The same experience is provided to both models in each setting.

DGO makes more effective use of relevant experience. As shown in Figure 4, adding relevant experience yields substantially larger gains for DGO. On Qwen3-4B-Base, the improvement is 28.3% for DGO, compared with 14.7% for GRPO; on Qwen3-8B-Base, the gain is 37.9% for DGO versus 18.9% for GRPO. This indicates that DGO is better at extracting useful signals from relevant experience.

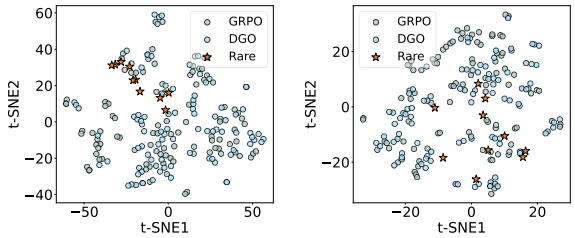
DGO is more robust to noisy experience. Injecting noisy experience leads to a noticeably smaller degradation for DGO than for GRPO on both model scales, and on Qwen3-8B-Base the performance of DGO is nearly unchanged. This suggests that DGO not only benefits more from helpful experience, but also resists distraction from irrelevant experience more effectively.

5.2 The Role of Experience in Internalization

We examine internalization through experience-free inference, where no external experience is available at test time. As shown in Table 1, DGO consistently outperforms GRPO under intrinsic inference, indicating improved standalone reasoning. Figure 3 shows that later iterations start from higher accuracy and maintain stronger performance, suggesting that experience-guided behaviors are partially transferred into the policy.

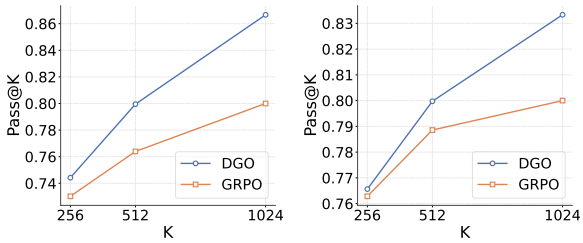
To further analyze these induced behaviors, we compare DGO and GRPO from two perspectives. First, we examine generated trajectories in a shared semantic space. Following prior work (Cheng et al., 2025), *rare trajectories* are defined as the top 10% of DGO trajectories farthest from the GRPO distribution, measured by the average distance to their k=5 nearest GRPO neighbors. Second, we use Pass@K to test whether these behaviors correspond to valid solution modes reachable during sampling (Yue et al., 2025).

As shown in Figure 5, DGO produces trajectories that are farther from the GRPO distribution, suggesting that experience-guided training induces reasoning behaviors beyond those commonly reached by standard RL. Figure 6 further shows that DGO achieves higher Pass@K across sampling budgets, indicating that these additional behaviors are not merely diverse but also contain valid solution modes. These results provide behavior-level evidence that DGO supports partial internalization of experience-induced reasoning patterns.



(a) Qwen3-8B-Base (b) Qwen3-4B-Base

Figure 5: t-SNE visualization of generated trajectories on AIME24 for the DGO- and GRPO-trained models.



(a) Qwen3-8B-Base (b) Qwen3-4B-Base

Figure 6: Comparison of the DGO- and GRPO-trained models on AIME24 in terms of Pass@K Accuracy.

6 Conclusion

In this paper, we propose **DGO**, a closed-loop training method that improves LLM reasoning through dual guidance from external and internal experience. DGO integrates experience construction, joint trajectory-policy refinement, and trajectory distillation into an iterative process, enabling the model to better utilize external experience and improve experience-free reasoning. Experiments on multiple benchmarks show that DGO improves both intrinsic reasoning and test-time scaling performance. Future work includes enabling models to manage and update their own experience more autonomously, and to coordinate external and internal experience more effectively. This may further improve the efficiency, robustness, and sustainability of experiential learning in LLMs.

Limitations

First, our experiments cover multiple model scales, including 4B, 8B, and 14B, and show consistent improvements across them. Experiments on even larger models would provide a complement. Due to computational resource constraints, we leave broader scaling studies to future work. Second, DGO is trained on mathematical reasoning data, and results on GPQA-Diamond and HLE-Verified provide initial evidence of transfer beyond the training domain. Since these evaluations still focus on relatively controlled reasoning settings, extending DGO to more heterogeneous or open-ended agentic domains, such as tool use, multi-step environment interaction, and long-horizon workspace tasks, remains a meaningful direction to be explored.

Ethics Statement

We strictly follow research ethics and use only publicly available datasets and models under their intended research usage. DGO is designed to improve the reasoning capabilities of LLMs through experience utilization and internalization in controlled RLVR settings. Our experiments focus on mathematical, scientific, and logical reasoning benchmarks, and do not involve human subjects, private user data, or personally identifiable information. AI assistance was used only during the paper refinement process.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. 2025. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. 2025. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. *arXiv preprint arXiv:2512.10696*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Jia Deng, Jie Chen, Zhipeng Chen, Daixuan Cheng, Fei Bai, Beichen Zhang, Yinqian Min, Yanzipeng Gao, Wayne Xin Zhao, and Ji-Rong Wen. 2025. From trial-and-error to improvement: A systematic analysis of llm exploration mechanisms in rlvr. *arXiv preprint arXiv:2508.07534*.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. 2025. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, and 8 others. 2026. Cl-bench: A benchmark for context learning. *Preprint*, arXiv:2602.03587.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling, 2023. *URL https://arxiv. org/abs/2308.08998*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. 2025. Skywork open reasoner series. Notion Blog.

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

Jingcheng Hu, Yinmin Zhang, Shijie Shang, Xiaobo Yang, Yue Peng, Zhewei Huang, Hebin Zhou, Xin Wu, Jie Cheng, Fanqi Wan, and 1 others. 2026. Pacore: Learning to scale test-time compute with parallel coordinated reasoning. *arXiv preprint arXiv:2601.05593*.

Shiting Huang, Zecheng Li, Yu Zeng, Qingnan Ren, Zhen Fang, Qisheng Su, Kou Shi, Lin Chen, Zehui Chen, and Feng Zhao. 2026. Internalizing meta-experience into memory for guided reinforcement learning in large language models. *arXiv preprint arXiv:2602.10224*.

Shashank Kirtania, Param Biyani, Priyanshu Gupta, Yasharth Bajpai, Roshni Iyer, Sumit Gulwani, and Gustavo Soares. 2025. Improving language agents through brew. *arXiv preprint arXiv:2511.20297*.

David A Kolb. 2014. *Experiential learning: Experience as the source of learning and development*. FT press.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The twelfth international conference on learning representations*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First conference on language modeling*.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. 2025. Meminsight: Autonomous memory augmentation for llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33124–33140.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. 2026. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*.

Hajime Shirouzu, Naomi Miyake, and Hiroyuki Masukawa. 2002. Cognitively active externalization for situated reflection. *Cognitive science*, 26(4):469–501.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, and 1 others. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*.

Sai Wang, Yu Wu, and Zhongwen Xu. 2025. Cogito, ergo ludo: An agent that learns to play by reasoning and planning. *arXiv preprint arXiv:2509.25052*.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, and 1 others. 2025. Bapo: Stabilizing off-policy reinforcement learning for llms via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and 1 others. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025b. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

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

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

Weiqi Zhai, Zhihai Wang, Jinghang Wang, Boyu Yang, Xiaogang Li, Xiang Xu, Bohan Wang, Peng Wang, Xingzhe Wu, Anfeng Li, and 1 others. 2026. Hle-verified: A systematic verification and structured revision of humanity's last exam. *arXiv preprint arXiv:2602.13964*.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. 2025. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. 2026. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Algorithm 1 DGO

Input: Training set $\mathcal{D}$, initial model $\pi_{\theta^{(0)}}$, pre-built experience bank $\mathcal{E}$, experience generator G, number of stages K, annealing schedule $\{\alpha_k\}_{k=1}^{K}$, internalization ratios $\{\beta_k\}_{k=1}^{K}$
Output: Final policy $\pi_{\theta_{\text{RL}}^{(K)}}$
$\theta_{\text{start}}^{(1)} \leftarrow \theta^{(0)}$
$\mathcal{S}_{\text{imp}}^{(\leq 0)} \leftarrow \emptyset, \mathcal{S}_e^{(\leq 0)} \leftarrow \emptyset$
for $k = 1$ to K **do**
  // Joint Trajectory-Policy Refinement
  Retrieve relevant experience from $\mathcal{E}$ and construct $\mathcal{D}_e^{(k)}$
  Construct mixed data with annealing coefficient α_k
  $\mathcal{D}_{\text{RL}}^{(k)} \leftarrow \alpha_k \mathcal{D}_e^{(k)} + (1-\alpha_k)\mathcal{D}_{\text{free}}^{(k)}$
  $\theta_{\text{RL}}^{(k)} \leftarrow \text{Update}(\theta_{\text{start}}^{(k)}, \mathcal{D}_{\text{RL}}^{(k)})$
  Collect rollout trajectories $\mathcal{S}^{(k)}$
  Update the experience bank $\mathcal{E}$ using $\mathcal{S}^{(k)}$ and G
  // Experience Internalization
  $\widetilde{\mathcal{S}}_e^{(k)} \leftarrow \text{Rewrite}(\mathcal{S}_e^{(k)})$
  $\mathcal{S}_{\text{imp}}^{(k)} \leftarrow \text{Filter}(\widetilde{\mathcal{S}}_e^{(k)})$
  $\mathcal{S}_{\text{imp}}^{(\leq k)} \leftarrow \mathcal{S}_{\text{imp}}^{(\leq k-1)} \cup \mathcal{S}_{\text{imp}}^{(k)}$
  $\mathcal{S}_e^{(\leq k)} \leftarrow \mathcal{S}_e^{(\leq k-1)} \cup \mathcal{S}_e^{(k)}$
  $\mathcal{D}_{\text{dist}}^{(k)} \leftarrow \beta_k \mathcal{S}_{\text{imp}}^{(\leq k)} + (1-\beta_k)\mathcal{S}_e^{(\leq k)}$
  $\theta_{\text{start}}^{(k+1)} \leftarrow \text{Distill}(\theta^{(0)}, \mathcal{D}_{\text{dist}}^{(k)})$
  // used to warm-start the next RL iteration
end for
return $\pi_{\theta_{\text{RL}}^{(K)}}$

A Pseudocode of DGO

We present the pseudocode of our method in Algorithm 1. At each iteration, the internalization step is used only to produce the initialization for the next RL round, while the final model used for evaluation is the RL policy obtained in the last iteration.

B Training Details

B.1 RL Stage

To ensure stability and effectiveness in our RL experiments, we adopt the following configuration. Please refer to Table 4. All training experiments are implemented on 32 × H20 GPUs.

B.2 Internalization Stage

We show the internalization training configurations in Table 5.

B.3 Experience Generator Training

We show the training configurations of experience generator in Table 6. The training data is sourced from SkyWork as well. Note that the training data does not overlap with the evaluation benchmarks or include test-set solutions.

Hyper-parameter	Value
Learning Rate	1e-6
Total Steps	200
Batch Size	128
Mini Batch Size	32
KL Loss Coefficient	0.0
Clip Higher	(0.28, 0.2)
Temperature	1.0
Number of Rollouts	16
Maximum Prompt Length	2K
Maximum Response Length	12K

Table 4: RL Training hyper-parameters.

Hyper-parameter	Value
Learning Rate	1e-6
Total Epochs	4
Batch Size	128 for 8B/14B; 64 for 4B
Maximum Length	6144

Table 5: SFT Training hyper-parameters.

C Additional Details

C.1 Rule of Experience Update

Experience update follows a redundancy-first, quality-second procedure:

- **Redundancy check.** For each new experience, we first compute its semantic similarity with the existing experiences in the bank. If the maximum similarity is below a threshold τ_{sim}, the new experience is treated as a candidate addition. Otherwise, it is regarded as redundant with the most similar existing experience and is treated as a candidate replacement. We use a predefined similarity threshold $\tau_{\text{sim}} = 0.8$ for redundancy detection.

- **Quality check.** We then evaluate the candidate action on the validation set using avg@16. If the new experience is a candidate addition, it is added only when it improves the validation performance of the current bank. If it is a candidate replacement, it replaces the old experience only when it yields a higher avg@16 than the old one.

C.2 Rule of Trajectory Filter

When rewriting experience-guided trajectories, strict filtering is applied to the rewritten trajectories. The filtering criteria include:

- trajectories with incorrect final answers;

Hyper-parameter	Value
Learning Rate	1e-5
Total Epochs	4
Batch Size	128
Maximum Length	16384

Table 6: Experience Generator Training hyper-parameters.

- trajectories that still explicitly rely on experience, as indicated by keywords such as 'refer to experience';

- trajectories that are excessively long, exceeding 6K in length;

- noisy trajectories containing long segments of meaningless strings.

C.3 Additional Experiments

C.3.1 Detailed TTS Results

To provide direct experience-utilization baselines beyond simple EGI, Table 7 evaluates different trained policies under the same TTS protocol. All methods use the same experience generator and the same number of TTS rounds, allowing a direct comparison of how effectively each policy exploits external experience during inference. Overall, DGO obtains the strongest average performance across model scales, indicating that its gains under TTS reflect improved experience utilization rather than merely access to external experience.

C.3.2 Efficiency Analysis

To assess the computational overhead introduced by DGO, we report the stage-wise training cost in GPU hours. As shown in Table 8, the dominant cost still comes from the RL rollout and policy update stage. Under the same number of RL training steps, DGO has a comparable RL-stage cost to GRPO and does not require additional RL rollout or optimization loops. Its extra cost comes only from three lightweight stages: experience renewal, trajectory rephrasing, and supervised distillation. Experience renewal summarizes collected trajectories into updated experience, trajectory rephrasing converts experience-guided trajectories into self-contained reasoning traces, and distillation performs short supervised learning on selected trajectories.

In total, these additional stages increase the cost from 1552 to 1784 GPU hours, *i.e.,* about a $1.15\times$ total cost relative to GRPO. This indicates that DGO introduces only a modest overhead beyond

Model	Method	In-domain				Out-of-domain		Avg
		AIME24	AIME25	HMMT25	MATH500	GPQA-D	HLE	
Qwen3-4B-Base	EGI (TTS)	6.88	4.79	2.29	39.60	16.35	2.02	11.99
	SFT (TTS)	24.79	16.25	10.42	72.92	46.38	4.25	29.17
	GRPO (TTS)	39.79	24.79	13.96	76.80	48.17	5.33	34.81
	DAPO (TTS)	34.58	25.00	13.96	77.12	50.69	**5.67**	34.50
	DGO (TTS)	**40.21**	**27.71**	**15.83**	**78.35**	**52.53**	4.45	**36.51**
Qwen3-8B-Base	EGI (TTS)	7.29	7.29	3.54	46.42	30.43	3.44	16.40
	SFT (TTS)	29.79	18.33	13.12	82.85	50.00	5.80	33.31
	GRPO (TTS)	39.17	25.42	14.17	80.38	52.97	6.46	36.43
	DAPO (TTS)	35.42	21.67	12.71	**85.25**	51.70	6.49	35.54
	DGO (TTS)	**49.38**	**30.63**	**14.37**	80.42	**54.99**	**6.51**	**39.38**
Qwen3-14B-Base	EGI (TTS)	13.96	9.79	6.67	63.32	39.84	4.85	23.07
	SFT (TTS)	31.04	25.62	17.08	85.35	43.18	6.74	34.84
	GRPO (TTS)	54.37	33.96	20.00	92.92	47.66	7.45	42.73
	DAPO (TTS)	46.67	31.67	19.79	91.75	**54.04**	7.30	41.87
	DGO (TTS)	**54.48**	**39.38**	**22.29**	**93.17**	51.64	**7.71**	**44.78**

Table 7: Detailed experience-utilization results under the same TTS protocol. All methods are evaluated with the same experience generator and number of TTS rounds, allowing a direct comparison of how different trained policies utilize external experience at inference time. **Bold** indicates the best result within each model, and underline indicates the second-best result.

Method	RL	Renew	Rephrase	Distill	Total
GRPO	1552	-	-	-	1552
DGO	1568	48	40	128	1784 ($\times$ 1.15)

Table 8: Training cost comparison in GPU hours (on Qwen3-8B-Base). DGO uses the same RL budget as GRPO and introduces additional lightweight stages, including experience renewal, trajectory rephrasing, and SFT distillation.

standard RL training. Despite this small additional cost, DGO brings consistent improvements under both intrinsic inference and experience-guided TTS. These results suggest that the extra computation is effectively spent on transforming collected trajectories into reusable experience and training signals, rather than simply scaling up RL rollout or optimization.

For inference, TTS introduces additional sampling cost proportional to the number of rounds, whereas intrinsic inference uses the same decoding budget as the corresponding non-TTS baselines. Therefore, the intrinsic gains reflect improved experience-free reasoning under an equal inference budget, while the TTS gains further measure the model's ability to exploit external experience when additional test-time sampling is allowed.

C.3.3 Sensitivity to α and β

The coefficients α and β control two key trade-offs in DGO. α determines the proportion of experience-guided instances during RL training, while β determines the mixture of rewritten implicit trajectories and experience-guided trajectories during distillation. Tables 10 and 11 report sensitivity results on AIME24 and AIME25 of Qwen3-4B-Base.

As shown in Table 10, the default annealing schedule $1.0 \rightarrow 0.5 \rightarrow 0.25$ achieves the best performance. Keeping α constant at 1.0 substantially hurts performance, indicating that excessive dependence on explicit experience guidance weakens internalization. A constant $\alpha = 0.5$ improves over constant 1.0 but remains worse than the annealed schedule, suggesting that DGO benefits from strong early guidance followed by reduced reliance on external experience.

Table 11 shows that $\beta = 0.5$ performs best, while $\beta = 0.25$ and $\beta = 0.75$ are slightly worse. This indicates that a balanced mixture of rewritten implicit trajectories and experience-guided trajectories is beneficial for distillation. The former encourages intrinsic reasoning, whereas the latter helps preserve experience utilization.

C.3.4 Full Component Ablation Results

The full component ablation results are shown in Table 9.

Method	In-domain (%)				Out-of-domain (%)		Avg	Avg Drop ↓
	AIME24	AIME25	HMMT25	MATH500	GPQA-D	HLE		
DGO (zero)	**27.08**	**21.04**	**9.17**	**70.08**	**40.47**	2.99	**28.47**	-
w/o *EA*	19.38	17.50	7.71	68.08	38.64	**3.24**	25.76	2.71
w/o *ER*	19.58	18.75	7.50	68.53	38.57	2.90	25.97	2.50
w/o *TR*	22.08	17.50	6.04	66.65	39.90	2.47	25.77	2.70
w/o *PD*	23.75	20.83	7.71	67.73	38.26	2.79	26.84	1.63
DGO (TTS)	40.83	**26.67**	**15.42**	**73.90**	**52.65**	**4.58**	**35.68**	-
w/o *EA*	37.71	24.58	13.12	70.43	48.30	3.37	32.92	2.76
w/o *ER*	**42.92**	23.75	15.21	70.78	48.42	3.11	34.03	1.65
w/o *TR*	31.87	20.42	11.67	72.10	48.11	3.99	31.36	4.32
w/o *PD*	34.58	22.71	15.00	67.00	49.31	3.20	31.97	3.71

Table 9: Component ablation study on Qwen3-4B-Base. *EA*, *ER*, *TR*, and *PD* denote *Experience Annealing*, *Experience Renewal*, *Trajectory Rephrasing*, and *Policy Distillation*, respectively. Avg Drop ↓ denotes the average-score decrease relative to the corresponding full DGO setting. Experiments are conducted between iteration 1 and 2.

α Schedule	AIME24	AIME25	Avg
$1.0 \rightarrow 0.5 \rightarrow 0.25$	**27.08**	**21.04**	**24.06**
constant 1.0	19.38	17.50	18.44
constant 0.5	23.71	19.65	21.68

Table 10: Sensitivity analysis of the experience annealing coefficient α. We fix $\beta = 0.5$ and vary the schedule of α, which controls the ratio of experience-guided instances during RL training.

β	AIME24	AIME25	Avg
0.25	25.92	18.30	22.11
0.50	**27.08**	**21.04**	**24.06**
0.75	26.11	20.84	23.48

Table 11: Sensitivity analysis of the distillation mixing ratio β. We use the default α schedule $1.0 \rightarrow 0.5 \rightarrow 0.25$ and vary β, which controls the mixture of rewritten implicit trajectories and experience-guided trajectories during distillation.

D Prompt Template

D.1 Prompt for Experience Extraction

Our prompt for experience extraction is shown in Table 12.

D.2 Prompt for Reasoning

D.2.1 Reasoning without Experience

Our prompt for reasoning without experience is shown in Table 13.

D.2.2 Reasoning with Experience

Our prompt for reasoning with experience is shown in Table 14.

D.3 Prompt for Rephrasing

Our prompt for solution rephrasing is shown in Table 15.

E Case Study

E.1 Example of Experience

We present several examples of experiences generated by our experience generator in Table 16.

E.2 Comparison of Utilization between DGO and GRPO

In Table 17, both the DGO-trained model and the GRPO-trained model are provided with the same retrieved experiences, all of which are clearly irrelevant to the current problem. The experiences concern set counting and inclusion-exclusion, whereas the target problem is a product over roots of unity. This setup therefore serves as a direct test of robustness to noisy retrieved experience.

The DGO-trained model remains largely problem-driven despite the presence of irrelevant experience. Its trajectory quickly identifies the algebraic structure of the expression, factorizes x^2-2x+2 into $(x-(1+i))(x-(1-i))$, and then applies the standard roots-of-unity product identity to transform the original product into evaluations at $1 \pm i$. The overall reasoning path is coherent and tightly aligned with the mathematical structure of the problem. Importantly, although noisy experiences are present in the prompt, they do not meaningfully alter the solution strategy. This suggests that the DGO-trained model is able to filter out irrelevant external guidance and rely on intrinsically relevant structure when the retrieved experience is unhelpful.

By contrast, the GRPO-trained model is more vulnerable to the same noisy experience. Although it begins with a superficially similar setup, it fails to maintain a stable problem-specific line of reasoning and eventually makes an invalid substitution, in-

correctly reducing the target product to $Q(2)$. This error reflects not simply a calculation mistake, but a weaker ability to discriminate between structurally relevant and irrelevant guidance. Under noisy retrieval, the model appears more prone to drifting toward a loose template-based manipulation rather than preserving the exact algebraic requirements of the problem. As a result, it reaches the wrong answer.

This comparison highlights an important difference in experience utilization. Stronger utilization is not merely the ability to use retrieved experience when it is helpful, but also the ability to reject or suppress it when it is irrelevant. The DGO-trained model demonstrates better robustness under noisy experience, whereas the GRPO-trained model is more easily distracted by irrelevant context. This suggests that DGO improves not only the effectiveness of leveraging useful experience, but also the selectivity required to avoid being misled by unhelpful experience.

E.3 Comparison of Internalization between DGO and GRPO

In Table 18, no external experience is provided at inference time, so the difference between the two trajectories mainly reflects the extent to which useful reasoning patterns have been internalized into the model parameters. The problem itself is structurally simple: after expanding the logarithms, the key step is to introduce $\log_x y$ and $\log_y x$ and then use the reciprocal relation between them. Once this relation is recognized, the quantity xy collapses immediately to a constant.

The DGO-trained model exhibits exactly this kind of internalized competence. Its trajectory quickly abstracts the problem into two compact symbolic relations, explicitly introduces auxiliary variables, and then invokes the identity $\log_y x = 1/\log_x y$ to connect the two equations. The subsequent derivation is short, stable, and fully symbolic, leading directly to $xy = 25$. Notably, the model does not rely on trial-and-error search or unnecessary algebraic expansion. This suggests that the relevant reasoning pattern has been internalized as a reusable parametric capability.

By contrast, the GRPO-trained model does not show the same level of internalization. Although it begins from the same logarithmic expansions, it fails to consolidate them into the reciprocal-log structure that makes the problem easy. Instead, it substitutes one equation into the other, produces a complicated self-referential expression, and then falls back to repeated numerical guessing. The trajectory becomes increasingly unstable and eventually loses focus altogether. This behavior indicates that the model has not fully absorbed the underlying simplification pattern into its parameters, even though the problem can in principle be solved by a short symbolic argument.

This comparison reveals the role of experience internalization. Strong internalization is reflected not only in getting the correct answer, but in whether the model can spontaneously recover the right abstraction without external guidance. The DGO-trained model demonstrates stronger internalization by directly activating a compact and transferable reasoning schema, whereas the GRPO-trained model is more likely to remain trapped in local manipulations without reaching the core structural insight.

E.4 Quality of Experience across Iterations

As shown in Table 19, across the three rounds, the extracted experience becomes progressively more structured, more actionable, and better aligned with the true computational bottleneck of the problem.

The Round 0 experience already captures the key algebraic reduction: the problem can be transformed into the divisibility condition $(b-1) \mid k(k-1)$ together with the formula $d_1 = \frac{k(k-1)}{b-1}$. This is a meaningful step beyond naive brute-force search, since it identifies the central arithmetic constraint underlying valid b-beautiful numbers. However, the experience is still relatively coarse-grained. It points to factoring $b-1$ and using divisibility, but does not yet describe a complete reasoning chain from the original digit condition to a systematic counting procedure.

The Round 1 experience is more complete and operational. Instead of starting from an already reduced form, it begins from the original condition $a + d = \sqrt{ab + d}$, explicitly introduces the substitution $k = a + d$, derives $a = \frac{k(k-1)}{b-1}$, and then reminds the solver to verify the digit constraints through $d = k - a$. Compared with Round 0, this version better preserves the full reasoning pipeline from the original statement to the final verification conditions. As a result, it is more directly reusable as a solution template rather than merely a high-level hint.

The Round 2 experience goes one step further by

making the counting structure itself more explicit. Rather than stopping at the divisibility condition, it turns that condition into a sharper congruence-based restriction on k, and uses it to narrow the candidate space to a much smaller set. This makes the experience more closely matched to the real objective of the problem, which is not just to verify a single candidate, but to identify the smallest base b for which the number of valid solutions exceeds a threshold. In this sense, the third-round experience is the most strategic: it abstracts away from local derivation details and focuses on how to reduce the search space efficiently.

Overall, the progression from Round 0 to Round 2 shows a clear refinement trend. The experience evolves from identifying a useful arithmetic condition, to providing a full derivation-and-checking template, and finally to exposing the higher-level structural principle that supports efficient counting. This illustrates how iterative refinement can gradually transform raw problem-solving traces into more compact, transferable, and strategically valuable experience.

```
### You are an expert knowledge extractor specializing in converting problem-solving trajectories into abstract, reusable
↪  reasoning rules that preserve crucial mathematical specificity and improve future problem solving efficiency.

Your task is to analyze a given problem and a candidate solution trajectory. Based on logical consistency, mathematical
↪  validity, and alignment with the problem requirements, judge whether the solution is likely correct or incorrect, and
↪  then extract **one concise general rule** as reusable experience.

The solution may be correct or incorrect. You must infer this by critically examining the reasoning and results.

#### Instructions:

1. **Input Analysis**:
    **problem**: The original problem statement.
    **solution**: A full solution trajectory attempting to solve the problem.

2. **Experience Extraction Guidelines**:
    - **Self-judgment**:
        Evaluate whether the solution is likely correct by checking:
        (a) whether the final answer satisfies the problem conditions, and
        (b) whether each reasoning step is logically and mathematically valid.

    - **If the solution appears correct**:
        Do NOT merely restate why it works.
        Instead, focus on extracting **optimization experience**, such as:
            · how the solution could be made shorter or more direct,
            · which intermediate steps, case splits, or calculations are redundant,
            · which key insight could be used earlier to avoid detours,
            · how to reformulate the approach into a cleaner standard method.
        Summarize this as a rule that helps future solvers reach the answer **faster and with less complexity**.

    - **If the solution appears incorrect**:
        Identify the most critical error, unjustified assumption, or logical gap.
        Extract an **avoiding-errors experience** that explains:
            · what warning sign to look for, and
            · what should be done instead to prevent this type of mistake.

    - **One Output**:
        Generate **exactly one rule** that reflects either:
        (a) an optimization rule distilled from a correct but improvable solution, or
        (b) an avoiding-errors rule distilled from an incorrect solution.

    - **Abstraction with Specificity**:
        Generalize the rule while retaining essential mathematical structures, formulas, constraints, or thresholds that
        ↪  make the experience precise and actionable.
        Use LaTeX for mathematical expressions where appropriate.

3. **Rule Format Requirement**:
    Output **exactly one rule** in the following structure, with no extra text:
        WHEN <Broad domain or phase>.
        IF <Specific condition, scenario, or inefficiency / potential error>.
        THEN <Action to take or avoid, emphasizing simplification, direct strategy, or error prevention, with key formulas
        ↪  if needed>.

# **Input:**
# problem:\n{problem}\n
# solution:\n{solution}\n

# **Output:**
WHEN <Broad domain or phase>. IF <Specific condition, scenario, or inefficiency / potential error>. THEN <Action to take
↪  or avoid>.
```

Table 12: Prompt template for experience extraction.

```
Please reason step by step, and put your final answer within \boxed{{}}.
{problem}
```

Table 13: Prompt template for reasoning without experience.

```
Please reason step by step, and put your final answer within \boxed{{}}. You will be given some reusable problem-solving
↪  experiences, and you can refer to them during the reasoning.

[Experiences]
{experiences}

[Problem]
{problem}
```

Table 14: Prompt template for reasoning with experience.

```
You are an expert mathematician. You are given a math problem, some background [Experience Content], and a [Reference
↪  Solution] which may have used those experiences explicitly.

Your task is to re-author a "Gold Standard" solution. This solution must be entirely self-contained, as if you solved the
↪  problem using only your innate mathematical intuition.

Strict requirements:

1. Total De-Reference: You MUST NOT mention, quote, or even subtly refer to the [Experience Content] provided. Do not use
↪  phrases like "By logic of...", "Following the principle of...", or any terminology that implies you are looking at an
↪  external guide. The reader should have no idea that any "Experience Content" was provided to you.

2. Invisible Integration: Instead of "using" the experience, you must "be" the expert who already knows it. If the
↪  experience suggests a shortcut or a formula, justify its use through first-principles reasoning within the flow of
↪  the solution. For example, do not say "Using the formula for X," say "Since the problem requires [reason], we can
↪  derive that [logical step]..."

3. Silent Audit & Correction: Treat the [Reference Solution] as a potentially flawed scratchpad. Verify every calculation
↪  and logical leap. If it is wrong, fix it silently. Your final output must be a clean, error-free, and authoritative
↪  proof. Do NOT mention that the reference was wrong.

4. Strategic Fidelity: Follow the general strategy of the [Reference Solution] (e.g., if it uses energy conservation, you
↪  should too), but execute it with perfect precision and without any meta-talk or conversational fillers.

5. Final Answer Consistency: The final numerical/symbolic answer must exactly match the [Reference Solution] and be
↪  enclosed in \boxed{{}}.

[Experience Content]
{experiences}

[Problem]
{problem}

[Reference Solution]
{solution}

[Your Solution]
```

Table 15: Prompt template for solution rewriting.

```
{
  "problem": "Let \(b\ge 2\) be an integer. Call a positive integer \(n\) \(b\text-\textit{eautiful}\) if it has exactly
  ↪  two digits when expressed in base \(b\) and these two digits sum to \(\sqrt n\). For example, \(81\) is
  ↪  \(13\text-\textit{eautiful}\) because \(81 = \underline{6}\ \underline{3}_{13}\) and \(6 + 3 = \sqrt{81}\). Find
  ↪  the least integer \(b\ge 2\) for which there are more than ten \(b\text-\textit{eautiful}\) integers.",
  "experiences": [
    "WHEN solving problems about numbers with given digit sums and square roots in a base representation.\nIF setting up
    ↪  an equation from the condition that the sum of digits equals the square root of the number,\nTHEN ensure the
    ↪  correct conversion from base \(b\) to decimal: if the two-digit representation is \(\overline{ad}_b\) with digits
    ↪  \(a\) (tens) and \(d\) (units), then \(n = ab + d\), and the condition is \(a + d = \sqrt{ab + d}\). Square both
    ↪  sides to get \((a+d)^2 = ab + d\), then rearrange to analyze integer solutions, noting digit constraints \(1 \le a
    ↪  \le b-1\), \(0 \le d \le b-1\). Avoid incorrectly writing \(n = b(a+1)\) which omits the units digit.",
    "WHEN solving for numbers with digit sum equal to the square root in a given base.\nIF the condition leads to \(n =
    ↪  k^2\) and \(d_1 = \frac{k(k-1)}{b-1}\) where \(d_1\) is the leading digit in base \(b\).\nTHEN use the
    ↪  divisibility condition \((b-1) \mid k(k-1)\) to find all \(k\) for each \(b\) by factoring \(b-1\), which is more
    ↪  efficient than brute-forcing over the range of \(k\).",
    "WHEN solving for the smallest base \( b \) with a specified count of numbers satisfying a digit-sum and square root
    ↪  condition.\nIF you have derived the relation \( d_1 = \frac{S(S-1)}{b-1} \) with \( S = d_1 + d_2 \), and \( b-1
    ↪  \le S^2 < b^2 \).\nTHEN iterate over integer \( S \ge 2 \), for each \( S \) compute all divisors \( d \) of \(
    ↪  S(S-1) \), set \( b = d+1 \), and check if \( b-1 \le S^2 < b^2 \); if so, record \( (b,S) \). Then collect all \(
    ↪  b \) and their counts to efficiently find the smallest \( b \) with the required count.",
    "WHEN solving problems about numbers with digit conditions in a given base.\nIF the condition leads to an equation of
    ↪  the form $d_1 = \frac{k^2 - k}{b-1}$ with $k = d_1 + d_2$ and $d_1$ must be an integer between $1$ and
    ↪  $b-1$,\nTHEN systematically check all integers $k$ from $\lceil \sqrt{b-1} \rceil$ to $b-1$, and verify that
    ↪  $(b-1) \mid k(k-1)$; this ensures a complete count without missing valid solutions."
  ]
}
```

Table 16: An example of retrieved experiences associated with a problem.

```
[Problem]
Let \omega\neq 1 be a 13th root of unity. Find the remainder when
\[
\prod_{k=0}^{12}(2-2\omega^k+\omega^{2k})
\]
is divided by 1000.

[Experiences]
[1] WHEN solving set-counting problems with multiple sets and given counts of exactly k-set intersections.
IF the total number of elements and the sizes of each set are known, and the goal is to find the intersection of all sets.
THEN compute the total number of set memberships \( T = \sum |A_i| \) and equate it to \( \sum_k k \cdot n_k \), where \(
↪  n_k \) is the number of elements in exactly \( k \) sets, leading to a linear equation for the unknown intersection
↪  count.

[2] WHEN solving overlapping set problems with given individual set sizes and counts of elements owning exactly two,
↪  exactly three, and all sets.
IF you need to find the number of elements owning all sets.
THEN compute total set memberships as the sum of individual set sizes, and set up the equation:
total memberships = (only one) + 2*(exactly two) + 3*(exactly three) + 4*(all four),
where only one = total elements - (exactly two + exactly three + all four); solve for all four.

[3] WHEN applying inclusion-exclusion to find the size of the intersection of all four sets.
IF you derive an equation by subtracting the total union size from the sum of individual set sizes.
THEN ensure the correct rearrangement:
sum singles - sum pairwise + sum triple - quadruple = union size,
so sum pairwise = sum singles + sum triple - quadruple - union size,
not sum singles - sum pairwise + sum triple = union size.
```

DGO-trained Trajectory (Correct)

```
When \(k=0\), the factor is
\[
2-2(1)+1^2=1,
\]
so it remains to evaluate
\[
\prod_{k=1}^{12}(2-2\omega^k+\omega^{2k}).
\]

Let
\[
P(x)=x^2-2x+2=(x-(1+i))(x-(1-i)).
\]

Then
\[
\prod_{k=1}^{12}P(\omega^k)
=
\left(\prod_{k=1}^{12}(\omega^k-(1+i))\right)
\left(\prod_{k=1}^{12}(\omega^k-(1-i))\right).
\]

Using
\[
\prod_{y^{13}=1,\ y\neq 1}(y-a)=\frac{a^{13}-1}{a-1},
\]
the product becomes
\[
[(1+i)^{13}-1][(1-i)^{13}-1].
\]

Now
\[
(1+i)^{13}=-64-64i,\qquad
(1-i)^{13}=-64+64i,
\]
so
\[
(-65-64i)(-65+64i)=8321.
\]

Hence,
\[
8321 \bmod 1000 = 321.
\]

\[
\boxed{321}
\]
```

GRPO-trained Trajectory (Incorrect)

```
Let
\[
f(y)=2-2y+y^2.
\]

We need
\[
P=\prod_{y\in\mu_{13}\setminus\{1\}} (2-2y+y^2).
\]

Consider
\[
Q(x)=\prod_{y\in\mu_{13}\setminus\{1\}}(x-y)
=\frac{x^{13}-1}{x-1}.
\]

Now, we can express \(P\) in terms of \(Q(x)\). Notice
↪  that
\[
P=Q(2).
\]

Thus
\[
P=2^{12}+2^{11}+\cdots+2+1=8191,
\]
so the remainder is
\[
8191 \bmod 1000=191.
\]

The trajectory later rewrites
\[
2-2y+y^2=(y-1)^2+1,
\]
but does not derive a valid product identity from it, and
still returns
\[
\boxed{191}.
\]
```

Table 17: Comparison of two trajectory excerpts under the same retrieved experiences. Both trajectories receive the same irrelevant experiences, but the left one stays problem-driven while the right one makes an invalid polynomial substitution.

```
[Problem]
There exist real numbers \(x\) and \(y\), both greater than 1, such that
\[
\log_x\left(y^x\right)=\log_y\left(x^{4y}\right)=10.
\]
Find \(xy\).
```

DGO-trained Trajectory (Correct)

```
From
\[
\log_x(y^x)=10,
\]
we get
\[
x\log_x y=10.
\]

Let
\[
a=\log_x y.
\]
Then
\[
x=\frac{10}{a}.
\]

From
\[
\log_y(x^{4y})=10,
\]
we get
\[
4y\log_y x=10.
\]

Let
\[
b=\log_y x.
\]
Then
\[
y=\frac{5}{2b}.
\]

Using the reciprocal relation
\[
\log_y x=\frac{1}{\log_x y},
\]
we have
\[
b=\frac{1}{a}.
\]

Thus
\[
y=\frac{5}{2\cdot (1/a)}=\frac{5a}{2}.
\]

Therefore,
\[
xy=\frac{10}{a}\cdot \frac{5a}{2}=25.
\]

\[
\boxed{25}
\]
```

GRPO-trained Trajectory (Incorrect)

```
From
\[
\log_x(y^x)=10,
\]
the solution rewrites
\[
y=x^{10/x}.
\]

From
\[
\log_y(x^{4y})=10,
\]
it rewrites
\[
x=y^{5/(2y)}.
\]

Then it substitutes one equation into the other:
\[
x=\left(x^{10/x}\right)^{5/(2x^{10/x})}
  =x^{25/(x^{10/x+1})},
\]
and concludes
\[
x^{10/x+1}=25.
\]

Instead of using the reciprocal relation between
\(\log_x y\) and \(\log_y x\), the trajectory then turns
↪  to
repeated numerical guessing and fails to obtain a stable
symbolic simplification.

The repeated trial steps are omitted here.

Final answer:
\[
\boxed{\frac{3}{10}}
\]
```

Table 18: Comparison of two trajectory excerpts without retrieved experience. The DGO-trained model directly invokes a compact logarithmic relation and reaches the correct result, while the GRPO-trained model fails to internalize the same simplification pattern and falls into unstable symbolic manipulation.

```
[Problem]
Let \(b\ge 2\) be an integer. Call a positive integer \(n\) \(b\text-\textit{eautiful}\) if it has exactly two digits
↪  when expressed in base \(b\) and these two digits sum to \(\sqrt n\). For example, \(81\) is
↪  \(13\text-\textit{eautiful}\) because \(81 = \underline{6}\ \underline{3}_{13}\) and \(6 + 3 = \sqrt{81}\). Find the
↪  least integer \(b\ge 2\) for which there are more than ten \(b\text-\textit{eautiful}\) integers.

[Round 0]
WHEN solving for numbers with digit sum equal to the square root in a given base.
IF the condition leads to \(n = k^2\) and \(d_1 = \frac{k(k-1)}{b-1}\) where \(d_1\) is the leading digit in base \(b\).
THEN use the divisibility condition \((b-1) \mid k(k-1)\) to find all \(k\) for each \(b\) by factoring \(b-1\), which is
↪  more efficient than brute-forcing over the range of \(k\).

[Round 1]
WHEN solving problems about digit conditions in a given base.
IF the condition is \( a+d = \sqrt{ab+d} \) with \( a \) the leading digit, \( 1 \le a \le b-1 \), \( 0 \le d \le b-1 \).
THEN set \( k = a+d \) so \( n = k^2 \), derive \( a = \frac{k(k-1)}{b-1} \), and find \( b \) such that \( b-1 \mid
↪  k(k-1) \) for multiple \( k \), ensuring \( a \) and \( d = k-a \) satisfy digit bounds, to avoid inefficient
↪  brute-force enumeration.

[Round 2]
WHEN solving problems about numbers with two digits in base \(b\) satisfying a digit-sum condition involving \(\sqrt{n}\).
IF you have derived that \(n = a b + d\) with \(a+d = k = \sqrt{n}\) and obtained the relation \(a = \frac{k^2 - k}{b-1}\).
THEN note that \(k(k-1)\) must be divisible by \(b-1\), which is equivalent to \(k \equiv 0\) or \(1 \pmod{b-1}\); use
↪  this congruence to restrict \(k\) to two arithmetic progressions within the range \(\lceil \sqrt{b} \rceil \le k \le
↪  b-1\), drastically reducing the number of candidates to check.
```

Table 19: The problem and one representative extracted experience from each round.