Training-Free Group Relative Policy Optimization

Anonymous authors
Paper under double-blind review

Abstract

Recent advances in Large Language Model (LLM) agents have demonstrated their promising general capabilities. However, their performance in specialized real-world domains often degrades due to challenges in effectively integrating external tools and specific prompting strategies. While methods like agentic reinforcement learning have been proposed to address this, they typically rely on costly parameter updates such as Supervised Fine-Tuning (SFT) or Group Relative Policy Optimization (GRPO) to alter output distribution. However, we argue that LLMs can achieve a similar effect on the output distribution by introducing a token prior, which is a far more lightweight approach that not only addresses practical data scarcity but also avoids the common issue of overfitting. To this end, we propose Training-free Group Relative Policy Optimization (Training-free GRPO), a cost-effective solution that enhances LLM agent performance without any parameter updates. Our method leverages minimal ground-truth data to perform multiple rollouts, where a group-based relative scoring mechanism is applied to iteratively distill high-quality experiential knowledge in each epoch. Such knowledge serves as the learned token prior, which is seamlessly integrated during LLM API calls to guide model behavior. Experiments on mathematical reasoning and web searching tasks demonstrate that Training-free GRPO, when applied to DeepSeek-V3.1, significantly improves out-of-domain performance. With just a few dozen training samples, Training-free GRPO outperforms fine-tuned small LLMs and achieves competitive results. Our code is available at `https://anonymous.4open.science/r/Training-Free-GRPO/`.

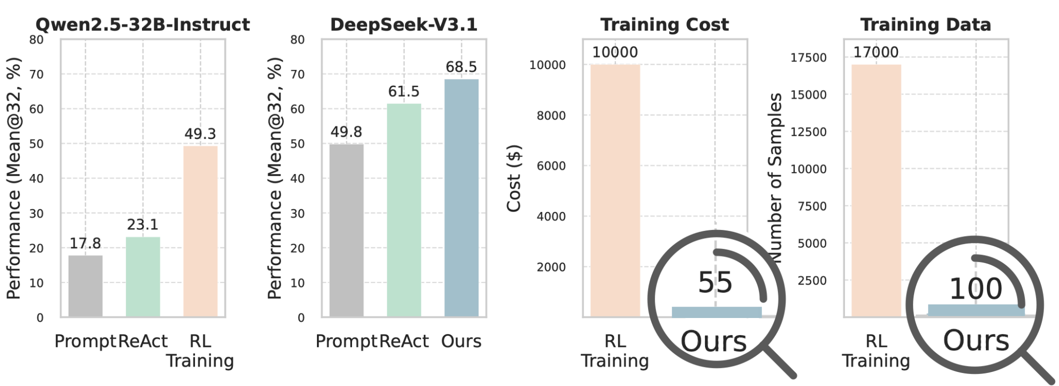


Figure 1: Training-free GRPO serves as a cost-effective alternative on AIME24 benchmarks.

1 Introduction

Large Language Models (LLMs) are emerging as powerful general-purpose agents capable of interacting with complex, real-world environments. They have shown remarkable capabilities across a wide range of tasks, including complex problem-solving (Feng et al., 2025a), advanced web research (Tongyi DeepResearch Team, 2025), code generation and debugging (Huang et al., 2023), and proficient computer use (Wang et al., 2024a). Despite their impressive capabilities, LLM agents often underperform in specialized, real-world domains. These scenarios typically demand the integration of external tools (e.g., calculators, APIs, databases), along with domain-specific task definitions and prompting strategies. Deploying a general-purpose agent out-of-the-box in such settings often results in suboptimal performance due to limited familiarity with domain-specific requirements or insufficient exposure to necessary tools.

To bridge this gap, agentic training has emerged as a promising strategy to facilitate the adaptation of LLM agents to specific domains and their associated tools. Recent advancements in agentic reinforcement learning (Agentic RL) approaches have employed Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and its variants to align model behaviors in the parameter space. Although these methods effectively enhance task-specific capabilities, their reliance on tuning LLM parameters poses several practical challenges:

- **Data Scarcity:** Fine-tuning LLMs typically necessitates large volumes of high-quality, carefully annotated data, which are often scarce and prohibitively expensive to obtain in specialized domains. Additionally, with limited samples, models are highly susceptible to overfitting, leading to poor generalization.
- **Computational Cost:** Even for smaller models, fine-tuning demands substantial computational resources, making it both costly and environmentally unsustainable. For larger models, the costs become prohibitive. Furthermore, fine-tuned models require dedicated deployment and are often limited to specific applications, rendering them inefficient for low-frequency use cases compared to more versatile general-purpose models.
- **Diminishing Returns:** The prohibitive training costs usually compel existing approaches to fine-tune smaller LLMs with fewer than 32 billion parameters, due to resource constraints rather than optimal design choices. While larger models would be preferred, the computational expense of fine-tuning necessitates this compromise. Paradoxically, API-based or open-source larger LLMs often deliver better cost-performance ratios through scalability and continuous model updates. However, these general-purpose models underperform in specialized domains where fine-tuning is necessary, creating a cost-performance dilemma.

Such limitations inherent in parameter tuning motivate a fundamental research question: *Is applying RL in parametric space the only viable approach? Can we enhance LLM agent performance in a non-parametric way with lower data and computational costs?*

We answer this question affirmatively by proposing **Training-free Group Relative Policy Optimization (Training-free GRPO)**, a novel and efficient method that improves LLM agent behavior in a manner similar to vanilla GRPO, while preserving the original model parameters unchanged. Our approach is motivated by the insight that LLMs can adapt their output distribution not only through parameter tuning, but also by leveraging a lightweight *token prior*, which encapsulates experiential knowledge obtained from a minimal set of training samples.

Training-free GRPO retains the multi-epoch rollout mechanism of vanilla GRPO, where multiple outputs are generated for each query to explore the policy space and evaluate potential strategies. While vanilla GRPO relies on gradient-based parameter updates to iteratively improve policy performance, Training-free GRPO eliminates this requirement by employing inference-only operations using LLMs. Rather than calculating a numerical advantage for gradient ascent within each group of rollouts, our method leverages LLMs to introspect on each group and distill a semantic advantage. This process requires multiple optimization iterations to progressively refine the policy, mirroring the iterative training structure of GRPO but without parameter updates. At each optimization step, this semantic advantage refines external experiential knowledge and guide policy outputs based on evolving contextual priors, thereby achieving policy optimization effects without modifying any model parameters.

By evaluating challenging mathematical reasoning and interactive web browsing tasks, we demonstrate that our method significantly enhances the performance of frozen policy models such as DeepSeek-V3.1 (DeepSeek AI, 2025) with only dozens of training samples. It outperforms fine-tuned models and achieves competitive results at a fraction of the computational cost, offering a simple and much more efficient alternative to traditional fine-tuning techniques.

Our principal contributions are threefold:

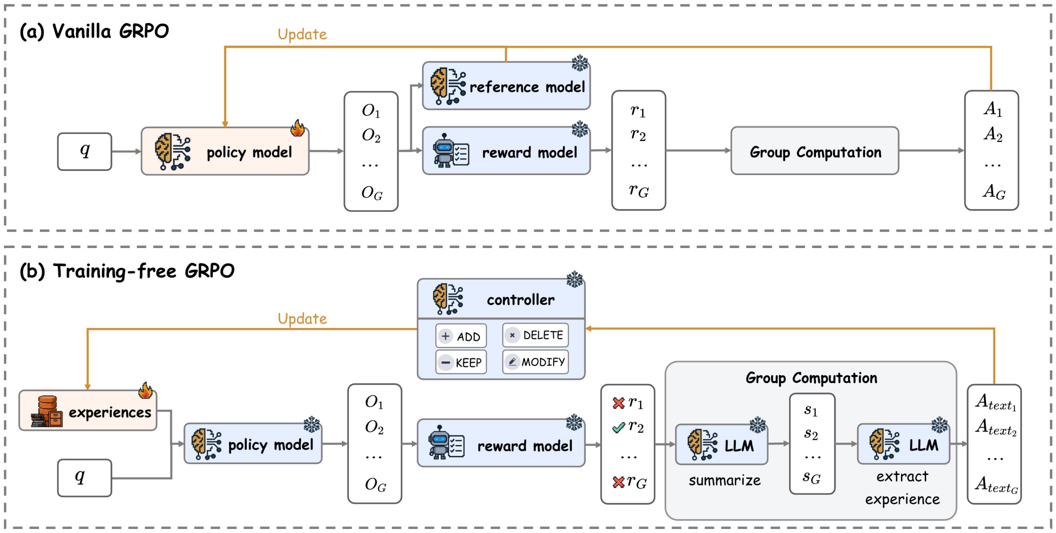


Figure 2: Comparison of Training-free GRPO and vanilla GRPO.

- **A New Training-free RL Paradigm:** We introduce Training-free GRPO, which shifts policy optimization from the parameter space to the context space by leveraging evolving token priors without gradient updates.
- **Semantic Group Advantage:** We replace numerical group advantage in vanilla GRPO with *semantic group advantage*, enabling LLMs to introspect their own rollouts and continuously updating experiential knowledge.
- **Data and Computational Efficiency:** Experiments confirm that Training-free GRPO effectively enhances the performance of a frozen policy with minimal training samples, offering a practical and cost-effective alternative across different domains.

2 TRAINING-FREE GRPO

In this section, we introduce our Training-free GRPO, a method designed to replicate the alignment benefits of the GRPO algorithm without performing any gradient-based updates to the policy model's parameters.

Vanilla GRPO. As shown in Figure 2, the vanilla GRPO procedure operates by first generating a group of G outputs $\{o_1, o_2, \dots, o_G\}$ for a given query q using the current policy LLM π_θ. Each output o_i is then independently scored with a reward model R. Subsequently, with rewards $\mathbf{r} = \{r_1, \dots, r_G\}$, it calculates a group-relative advantage $\hat{A}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$ for each output o_i. By combining a KL-divergence penalty against a reference model, it constructs a PPO-clipped objective function $\mathcal{J}_{\text{GRPO}}(\theta)$, which is then maximized to update the LLM parameters θ.

Training-free GRPO repurposes the core logic of this group-based relative evaluation but translates it into a non-parametric, inference-time process. Instead of updating the parameters θ, we leave θ permanently frozen and maintain an external *experience library* $\mathcal{E}$, which is initialized to $\emptyset$.

Rollout and Reward. As shown in Figure 2, our rollout and reward process mirrors that of GRPO exactly. Given a query q, we perform a parallel rollout to generate a group of G outputs $\{o_1, o_2, \dots, o_G\}$ using the LLM. Notably, while GRPO uses the current trainable policy π_θ, our policy conditions on the experience library, $\pi_\theta(\cdot|q, \mathcal{E})$. Identical to the standard GRPO setup, we score each output o_i by the reward model R to obtain a scalar reward $r_i = R(q, o_i)$.

Group Advantage Computation. To provide an optimization direction for policy parameters, vanilla GRPO computes a numerical advantage $\hat{A}_i$ that quantifies each output o_i's relative quality within its group. Similarly, Training-free GRPO performs an analogous comparison within each group but produces a *semantic advantage* in the form of natural language experience. Since $\hat{A}_i = 0$

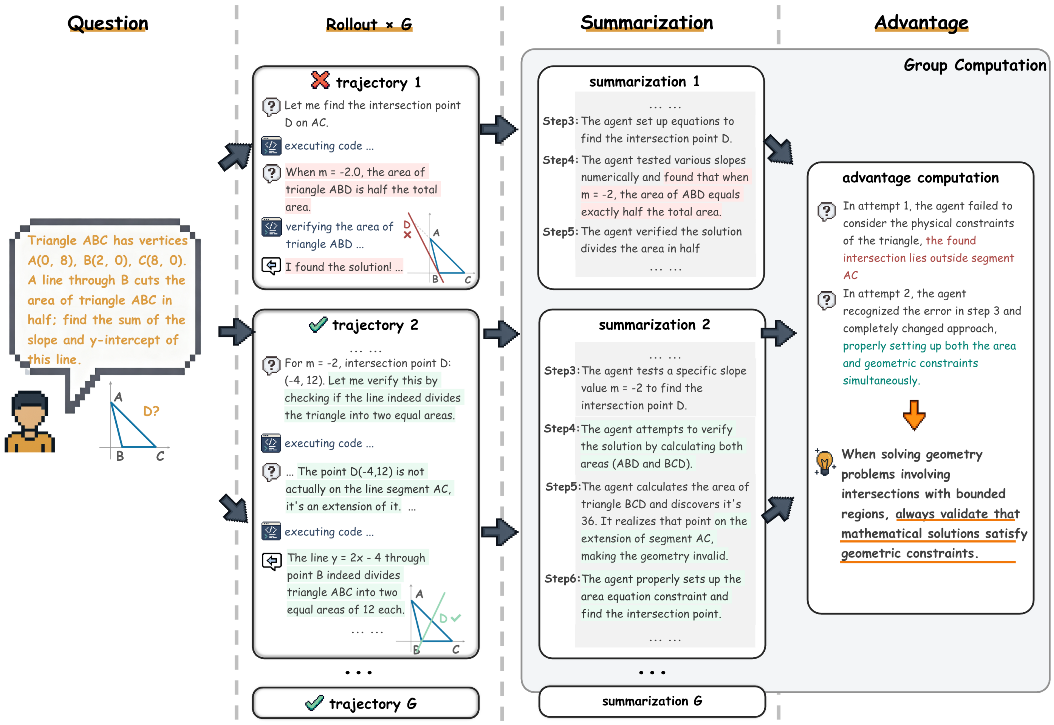


Figure 3: Example of a Training-free GRPO learning step.

when all G outputs receive identical rewards (i.e., $\text{std}(\mathbf{r}) = 0$) in vanilla GRPO, we generate such semantic advantages only for groups with both clear winners and losers. Specifically, for each output o_i, we first ask the same LLM $\mathcal{M}$ to provide a corresponding summary $s_i = \mathcal{M}(p_{\text{summary}}(q, o_i))$ separately, where p_{summary} is a prompt template that incorporates the query q and output o_i to form a structured summarization request. Given the summaries $\{s_1, s_2, \dots, s_G\}$ and the current experience library $\mathcal{E}$, the LLM $\mathcal{M}$ articulates the reasons for the relative success or failure of the outputs, followed by extracting a concise natural language experience $A_{\text{text}} = \mathcal{M}(p_{\text{extract}}(q, s_i, \mathcal{E}))$, where p_{extract} is another prompt template for experience extraction. This natural language experience A_{text} serves as our semantic advantage, functionally equivalent to vanilla GRPO's $\hat{A}_i$, encoding the critical experiential knowledge of what actions lead to high rewards.

Optimization. Whereas vanilla GRPO updates its model parameters θ via gradient ascent on $\mathcal{J}_{\text{GRPO}}(\theta)$ computed by all advantages in a single batch, we update our experience library $\mathcal{E}$ using all semantic advantages A_{text} from the current batch. Specifically, given the existing experiences library $\mathcal{E}$, we prompt the LLM to generate a list of operations based on all these A_{text}, where each operation could be: (1) *Add:* Directly append the experience described in A_{text} to the experience library $\mathcal{E}$. (2) *Delete:* Based on A_{text}, remove a low-quality experience from the experience library $\mathcal{E}$. (3) *Modify:* Refine or improve an existing experience in the experience library $\mathcal{E}$ based on insights from A_{text}. (4) *Keep:* The experience library $\mathcal{E}$ remains unchanged. After updating the experience library $\mathcal{E}$, the conditioned policy $\pi_\theta(y|q, \mathcal{E})$ produces a shifted output distribution in subsequent batches or epochs. This mirrors the effect of a GRPO policy update by steering the model towards higher-reward outputs, but achieves this by altering the context rather than the model's fundamental parameters. The frozen base model π_θ acts as a strong prior, ensuring output coherence and providing a built-in stability analogous to the KL-divergence constraint in GRPO that prevents the policy from deviating excessively from π_{ref}.

3 Evalution

To compare Training-free GRPO with competitive baselines, we conduct comprehensive experiments on both tool-integrated mathematical reasoning and web searching benchmarks.

Table 1: Mean@32 on AIME 2024 and AIME 2025 (%).

Type	Method	Model	Tool	AIME24	AIME25
Prompt	-	*Qwen2.5-32B-Instruct*	-	21.8	17.8
RL	GRPO	*Qwen2.5-32B-Instruct*	-	40.0	36.7
Prompt	ReAct	*Qwen2.5-32B-Instruct*	CI	29.6	23.1
RL	ReTool	*Qwen2.5-32B-Instruct*	CI	67.0	49.3
RL	SimpleTIR	*Qwen2.5-32B-Instruct*	CI	59.9	49.2
RL	ZeroTIR	*Qwen2.5-32B-Instruct*	CI	56.7	33.3
RL	AFM	*Qwen2.5-32B-Instruct*	CI	66.7	59.8
Prompt	-	*DeepSeek-V3.1*	-	66.3	49.8
Prompt	ReAct	*DeepSeek-V3.1*	CI	74.8	61.5
RL	Training-free GRPO	*DeepSeek-V3.1*	CI	**79.6**	**68.5**

3.1 Tool-integrated Math Reasoning

We first evaluate Training-free GRPO on challenging mathematical problem-solving tasks that require a code interpreter (CI) tool.

Benchmarks. We conduct our evaluation on the challenging AIME24 and AIME25 benchmarks (AIME, 2025), which are representative of complex, out-of-domain mathematical reasoning challenges. To ensure robust and statistically reliable results, we evaluate each problem with 32 independent runs and report the average Pass@1 score, which we denote as Mean@32.

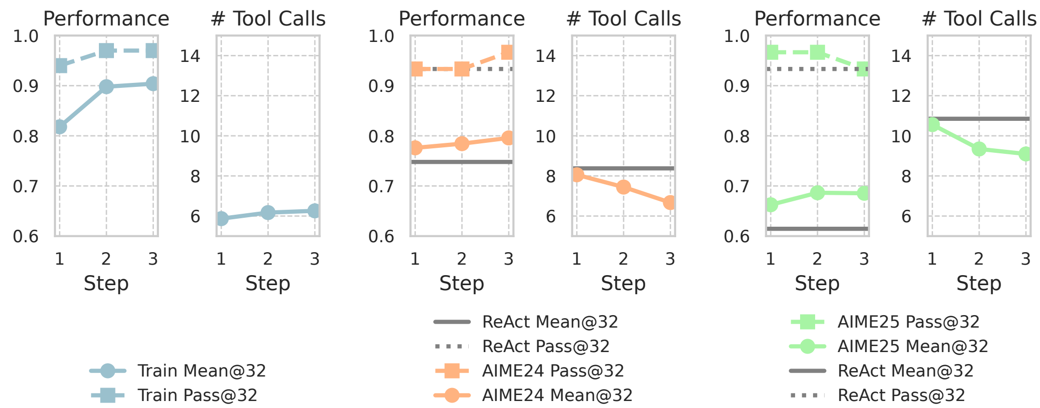
Methods. We compare several competitive LLMs of varying scales, including Qwen2.5-32B-Instruct (Qwen et al., 2025) and DeepSeek-V3.1 (DeepSeek AI, 2025). For each model, we evaluate two primary configurations: (1) Direct Prompting (a text-only input/output process), and (2) ReAct (Yao et al., 2023) with a code interpreter tool. We also present results from recent RL methods that incorporate tool-use capabilities, including ReTool (Feng et al., 2025a), SimpleTIR (Xue et al., 2025), ZeroTIR (Mai et al., 2025), and AFM (Li et al., 2025b). For our Training-free GRPO experiments, we randomly sample 100 problems from the DAPO-Math-17K dataset (Yu et al., 2025) and run the learning process for 3 epochs with a single batch per epoch. We use a temperature of 0.7 and a group size of 5 during the Training-free GRPO learning phase, and a temperature of 0.3 for the final evaluation on the AIME 2024 and 2025 benchmarks.

Main Results. As detailed in Table 1, our approach demonstrates remarkable improvement in mathematical reasoning. Based on DeepSeek-V3.1 with ReAct, already establishes a strong baseline, achieving 74.8% on AIME24 and 61.5% on AIME25. This performance surpasses various state-of-the-art RL methods like ReTool and AFM, which are fine-tuned on the smaller 32B-scale model. This observation highlights the inherent limitations of smaller models, even with extensive parameter-based training.

Applying Training-free GRPO to the frozen DeepSeek-V3.1 model elevates its performance to 79.6% on AIME24 and 68.5% on AIME25. This represents a significant absolute gains of +4.8% and +7.0% respectively, achieved with only 100 training examples and zero gradient updates. This result clearly demonstrates that guiding a powerful, frozen model through context-space optimization is a more effective and efficient strategy than exhaustively fine-tuning a less capable model.

Learning Dynamics. During the 3-step learning process on the training data, we observe a steady and significant improvement in both Mean@5 (from 81.8% to 90.4%) and Pass@5 (from 94% to 97%). Concurrently, the average number of tool calls increases from 5.9 to 6.3, indicating that as the agent refines its experiences at each step, it learns to utilize the code interpreter more effectively to solve training problems, validating the effectiveness of our semantic advantage guided optimization.

Furthermore, as shown in Figure 4, the Mean@32 performance on both AIME24 and AIME25 generally improves with each step, demonstrating that the learned experiences from only 100 DAPO-Math-17K problems generalize effectively and the necessity of multi-step learning. More strikingly, the average number of tool calls decreases during AIME evaluation, from 8.4 to 6.7 for AIME24 and from 10.9 to 9.1 for AIME25. This suggests that Training-free GRPO does not simply encourage correct reasoning and action; it also teaches the agent to use tools more efficiently and judiciously.



(a) Train set (b) AIME24 (c) AIME25

Figure 4: Detailed performance at each Training-free GRPO step.

Table 2: Mean@32 of Training-free GRPO with DeepSeek-V3.1 (%).

Type	Method	AIME24	AIME25
Prompt	ReAct	74.8	61.5
Prompt	ReAct + Self-generated Experiences	78.0	65.1
RL	Training-free GRPO without Ground Truths	79.1	67.7
RL	Training-free GRPO	**79.6**	**68.5**

Such learned experiential knowledge helps the agent prune suboptimal reasoning paths and avoid erroneous or redundant tool calls, leading to more direct and robust solutions.

Effectiveness of Learned Experiences. In Table 2, we also include the ReAct enhanced with the experiences generated by DeepSeek-V3.1, matching the quantity learned from Training-free GRPO. Interestingly, we find that such self-generated experiences also significantly boost the performance on both benchmarks. However, the experiences learned through Training-free GRPO consistently outperform them, achieving performance gains of +1.6% and +3.4% on AIME24 and AIME25, respectively. This highlights the superior quality and effectiveness of the Training-free GRPO for enhancing reasoning capabilities and overall performance.

Robustness to Reward Signal. Table 2 also presents a variant of Training-free GRPO, where the ground truth answers are not provided. In such cases, the semantic advantage is directly obtained by comparing the rollouts within each group, where the LLM can only rely on implicit majority voting, self-discrimination and self-reflection. Although it does surpass the default version with ground truths, Training-free GRPO still achieves an impressive results of 79.1% on AIME24 and 67.7% on AIME25. Thus, Training-free GRPO demonstrates robustness and applicability to domains where ground truths are scarce or unavailable, further broadening its practical utility.

3.2 Web Searching

In this section, we evaluate the effectiveness of Training-free GRPO in addressing web searching tasks by leveraging minimal experiential data to enhance agent behavior.

Datasets. For training, we constructed a minimal training set by randomly sampling 100 trajectories from the **AFM (Chain-of-Agents)** web interaction RL dataset (Li et al., 2025a). AFM provides high-quality, multi-turn interactions between agents and web environments, collected via reinforcement learning in realistic browsing scenarios. For evaluation, we employ **WebWalkerQA** benchmark (Wu et al., 2025), a widely-used dataset for assessing web agent performance. Its tasks require understanding both natural language instructions and complex web page structures, making it a rigorous evaluation framework for generalist agents.

Table 3: pass@1 on WebWalkerQA.

Type	Method	Model	pass@1
Prompt	ReAct	*DeepSeek-V3.1*	63.4
RL	Training-free GRPO	*DeepSeek-V3.1*	**67.1**

Table 4: Ablation results on WebWalkerQA subset.

Setting	Model	pass@1	pass@3
ReAct	*DeepSeek-V3.1*	66.67	72.55
ReAct	*QwQ-32B*	27.45	43.14
Random Experiences	*DeepSeek-V3.1*	68.63	78.43
Training-free GRPO w/o GT	*DeepSeek-V3.1*	70.59	74.51
Training-free GRPO	*QwQ-32B*	25.49	45.10
Training-free GRPO	*DeepSeek-V3.1*	**74.51**	**84.31**

Methods. Our proposed Training-free GRPO is applied to DeepSeek-V3.1 model without any gradient-based updates. We perform 3 epochs of training-free optimization with a group size of $G = 3$. The temperature settings follow those used in prior mathematical experiments.

Main Results. We evaluate the effectiveness of our proposed Training-free GRPO method on the WebWalkerQA benchmark. As shown in Table 3, our method achieves a pass@1 score of **67.1%** when using DeepSeek-V3.1 model, demonstrating significant performance improvements. Notably, this surpasses the baseline prompt-only approach with the same model, which achieves 63.4%, indicating that our approach effectively steers model behavior through learned experiential knowledge rather than relying solely on static prompting strategies.

Ablation. We conduct ablation studies on a stratified random sample of 51 instances from the WebWalkerQA test set, where the sampling is proportionally stratified by difficulty level to ensure balanced representation across different levels of complexity. All ablated models are evaluated after 2 epochs of experience optimization. The results are summarized in Table 4.

Using random experiences only slightly improves over ReAct (68.63% vs. 66.67%), confirming that mere in-context examples offer limited gains, which enhances the quality and strategic consistency of the experiences extracted by our method. Training-free GRPO without ground truth reaches 70.59%, demonstrating—consistent with math results—that relative reward evaluation remains effective even without ground truth.

Applying Training-free GRPO to QwQ-32B (Team, 2025) yields only 25.49% pass@1, significantly lower than the 74.51% achieved with DeepSeek-V3.1, and even underperforming its own ReAct baseline (27.45%). This may suggest that the effectiveness of our method is dependent on the underlying model's reasoning and tool-use capabilities, indicating that model capability is a prerequisite for effective experience-based optimization

4 Comparing RL on Context Space and Parameter Space

4.1 Cross-domain Transfer Analysis

A critical advantage of Training-free GRPO lies in its ability to achieve strong performance across diverse domains without suffering from the domain specialization trade-off observed in parameter-tuned methods. As demonstrated in Table 5, we observe a striking pattern of performance degradation when domain-specialized models are transferred to different domains. For instance, ReTool specifically trained on mathematical reasoning tasks, achieves competitive performance of 67.0% on AIME24 within its specialized domain. However, when transferred to web navigation tasks on WebWalker, its performance drops dramatically to only 18.3%. Similarly, MiroThinker optimized for web interactions, demonstrates strong web capabilities but achieves only 36.5% on AIME24. This significant performance drop highlights that parameter-based specialization narrows the model's ca-

Table 5: Cross-domain transfer performance comparison (pass@1, %).

Method	Training Domain	AIME24	AIME25	WebWalker
ReTool *(Qwen2.5-32B-Instruct)*	Math	67.0	49.3	18.3
MiroThinker *(Qwen3-32B)*	Web	36.5	29.6	53.6
Training-free GRPO *(DeepSeek-V3.1)*	Math / Web	**79.6**	**68.5**	**67.1**

pabilities to excel in the training domain at the expense of generalizability. In contrast, Training-free GRPO applied to the frozen LLM achieves state-of-the-art performance in both domains simultaneously by simply plugging in domain-specific learned experiences. Such cross-domain robustness makes Training-free GRPO particularly valuable for real-world applications where agents must operate in multifaceted environments with diverse requirements.

4.2 Computational Costs

As shown in Figure 1, we further analyze the economic advantages of Training-free GRPO by comparing its computational costs with a vanilla training approach, specifically ReTool, on mathematical problem-solving tasks. This comparison highlights the practical benefits of our method in scenarios characterized by limited data, constrained budgets, or volatile inference demand.

Training Cost. Using the same dataset and experimental setup, we replicate the training process of ReTool on Qwen2.5-32B-Instruct. The training requires approximately 20 thousand NVIDIA A100 GPU hours. Assuming a conservative rental price of $0.5 per GPU hour, the total training expense amounts to roughly $10,000. In contrast, Training-free GRPO, when applied to DeepSeek-V3.1, achieves superior performance on the same evaluation benchmarks (Section 3.1) while requiring only minimal fine-tuning. It requires only 3 training steps over 100 samples completed within 6 hours, which consumes 77 million input tokens and 6 million output tokens, amounting to a total cost of approximately $55 based on the official DeepSeek AI pricing[1]. The drastic reduction in training cost by over two orders of magnitude makes our approach especially cost-effective.

Inference Cost. Deploying a trained model like ReTool-32B entails significant fixed infrastructure costs. In a typical serving setup, 4 NVIDIA A100 GPUs with vLLM-based batching requests can process about 400 problems per hour from the AIME24 and AIME25 benchmarks. At the same $0.5 per GPU-hour pricing, the inference cost per problem averages $0.005. While this per-instance cost is relatively low, it presupposes continuous GPU availability, which becomes inefficient under fluctuating or low request volumes. In contrast, Training-free GRPO incurs a token-based cost. On average, each request consumes 82 thousand input tokens and 6 thousand output tokens, totaling about $0.06 per problem. Although per-query inference with a large API-based model is more expensive than with a dedicated small model, many real-world applications, particularly specialized or low-traffic services, experience irregular and modest usage patterns. In such cases, maintaining a dedicated GPU cluster is economically unjustifiable. By leveraging the shared, on-demand infrastructure of large model services like DeepSeek, Training-free GRPO eliminates fixed serving overhead and aligns costs directly with actual usage. This pay-as-you-go model is distinctly advantageous in settings where demand is unpredictable or sparse.

5 Related Work

LLM Agents. Large Language Models (LLMs) can overcome inherent limitations, such as lacking real-time knowledge and precise computation, by leveraging external tools. This has spurred the development of LLM agents that interleave reasoning with actions. Foundational frameworks like ReAct (Yao et al., 2023) prompt LLMs to generate explicit chain-of-thought (CoT) and actionable steps, enabling dynamic planning through tool use. Furthermore, Toolformer (Schick et al., 2023) demonstrates that LLMs can learn to self-supervise the invocation of APIs via parameter fine-tuning. Building on these principles, subsequent research has produced sophisticated single- and multi-agent systems, such as MetaGPT (Hong et al., 2024), CodeAct (Wang et al., 2024b), and OWL (Hu et al., 2025), which significantly enhance the quality of planning, action execution, and tool integration.

[1] `https://api-docs.deepseek.com/quick_start/pricing`

Reinforcement Learning. Reinforcement learning (RL) has proven highly effective for aligning large language models (LLMs) with complex, long-horizon goals. Foundational algorithms like Proximal Policy Optimization (PPO) (Schulman et al., 2017) employ a policy model for generation and a separate critic model to estimate token-level value. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates the need for a critic by estimating advantages directly from groups of responses. Recent research try to apply RL to transform LLMs from passive generators into autonomous agents that learn through environmental interaction. GiGPO (Feng et al., 2025b) implements a two-level grouping mechanism for trajectories, enabling precise credit assignment at both the episode and individual step levels. ReTool (Feng et al., 2025a) uses PPO to train an agent to interleave natural language with code execution for mathematical reasoning. Chain-of-Agents (Li et al., 2025b) facilitates multi-agent collaboration within a single model by using dynamic, context-aware activation of specialized tool and role-playing agents. Furthermore, Tongyi Deep Research (Tongyi DeepResearch Team, 2025) introduces synthetic high-quality data generation pipeline and conduct customized on-policy agentic RL framework. However, such parameter-updating approaches results in prohibitive computational cost, which typically restricts application to LLMs with fewer than 32B parameters. Also, they only achieve diminishing returns compared to simply using larger, more powerful frozen LLMs. In contrast, our proposed Training-free GRPO method seeks to achieve comparable training benefits on state-of-the-art LLMs without any parameter updates, drastically reducing both data and computational requirements.

Training-free Methods. A parallel line of research aims to improve LLM behavior at inference time without updating model weights. The general idea is in-context learning (ICL) (Brown et al., 2020), which leverages external or self-generated demonstrations within a prompt to induce desired behaviors. More recent methods introduce iterative refinement mechanisms. Self-Refine (Madaan et al., 2023) generates an initial output and then uses the same LLM to provide verbal feedback for subsequent revisions. Similarly, Reflexion (Shinn et al., 2023) incorporates an external feedback signal to prompt the model for reflection and a new attempt. In-context reinforcement learning (ICRL) (Song et al., 2025; Monea et al., 2024) demonstrates that LLMs can learn from scalar reward signals by being prompted with their past outputs and associated feedback. TextGrad (Yuksekgonul et al., 2025) proposes a more general framework, treating optimization as a process of back-propagating textual feedback through a structured computation graph. A key characteristic of these methods is their focus on iterative, within-sample improvement for a single query. In contrast, our Training-free GRPO more closely mirrors traditional RL by learning from a separate dataset across multiple epochs to iteratively refine a shared, high-quality experience library for all queries. Furthermore, given each query, unlike self-critique or context updates for a single trajectory, our method explicitly compares multiple rollouts per query, producing a semantic advantage to compare different trajectories in each group. Such semantic feedback is accumulated into a concise knowledge base, enabling a non-parametric analogue of policy improvement that biases the model's output distribution during inference. More specifically for optimizing agent systems, Agent KB (Tang et al., 2025) constructs a shared, hierarchical knowledge base to enable the reuse of problem-solving experiences across tasks. Unlike the complex reason-retrieve-refine process of Agent KB, Training-free GRPO simply injects the learned experience into the prompt. Moreover, Agent KB relies on hand-crafted examples and employs an off-policy learning paradigm once by collecting trajectories in the different way of online inference. In contrast, our Training-free GRPO uses a consistent pipeline and more closely mirrors on-policy RL with multi-epoch learning.

6 Conclusion

In this paper, we introduced Training-free GRPO, a novel paradigm that reframes policy optimization as a context-engineering problem rather than a parameter-updating one. By leveraging group-based rollouts to iteratively distill a semantic advantage into an evolving token prior, our method successfully steers the output distribution of a frozen LLM agent, achieving significant performance gains in specialized domains. Experiments demonstrate that this Training-free approach not only surmounts the practical challenges of data scarcity and high computational cost but also rivals the effectiveness of traditional fine-tuning. Our work establishes a new, highly efficient pathway for adapting powerful LLM agents, making advanced agentic capabilities more accessible and practical for real-world applications.

Ethics Statement The present study conforms with the ICLR Code of Ethics. The paper does not involve crowdsourcing nor research with human subjects.

Reproducibility Statement All datasets used in the paper are publicly accessible (see Section 3). All the codes are available at `https://anonymous.4open.science/r/Training-Free-GRPO` for reproduction.

References

AIME. Aime problems and solutions, 2025. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

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

DeepSeek AI. DeepSeek-V3.1 Release. DeepSeek API Docs, available at `https://api-docs.deepseek.com/news/news250821`, August 2025. Accessed on September 24, 2025.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025b.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. International Conference on Learning Representations, ICLR, 2024.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. OWL: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL `https://arxiv.org/abs/2505.23885`.

Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. 2025a. URL `https://arxiv.org/abs/2508.13167`.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-Agents: End-to-end agent foundation models via multi-agent distillation and agentic rl, 2025b. URL `https://arxiv.org/abs/2508.13167`.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Xinji Mai, Haotian Xu, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.

Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. Llms are in-context reinforcement learners. 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. *arXiv preprint arXiv:2302.04761*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Kefan Song, Amir Moeini, Peng Wang, Lei Gong, Rohan Chandra, Yanjun Qi, and Shangtong Zhang. Reward is enough: Llms are in-context reinforcement learners. *arXiv preprint arXiv:2506.06303*, 2025.

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

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL `https://qwenlm.github.io/blog/qwq-32b/`.

Tongyi DeepResearch Team. Tongyi-deepresearch. `https://github.com/Alibaba-NLP/DeepResearch`, 2025.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024b.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal. 2025. URL `https://arxiv.org/abs/2501.07572`.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

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. *arXiv preprint arXiv:2503.14476*, 2025.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

A Prompts for Math Tasks

Solve the following problem step by step. You now have the ability to selectively write executable Python code to enhance your reasoning process, e.g., calulating numbers and verifying math computations. Never directly just printing your semantic reasoning in Python. The Python code will be executed by an external sandbox, and the output (returned as a dict with the message in the "message" field) can be returned to aid your reasoning and help you arrive at the final answer. The Python code should be complete scripts, including necessary imports.

Each code snippet is wrapped with
```python
code snippet
```.

The last part of your final response should be in the following format:
<answer> \boxed{The final answer goes here.} </answer>

Figure 5: System prompt for math tasks.

Please solve the problem:
`{problem}`

When solving problems, you MUST first carefully read and understand the helpful instructions and experiences:
`{experiences}`

Figure 6: Prompt for supplementing math problems with experiential knowledge $\mathcal{E}$.

An agent system may be provided with some experiences, and then it produces the following trajectory to solve the given problem. Please summarize the trajectory step-by-step:
1. For each step, describe what action is being taken, and which experience has been used in this step.
2. Given the grading of this rollout and the correct answer, identify and explain any steps that represent detours, errors, or backtracking, highlighting why they might have occurred and what their impact was on the trajectory's progress.
3. Maintain all the core outcome of each step, even if it was part of a flawed process.

<trajectory> `{trajectory}` </trajectory>
<evaluation> `{whether the answer is correct or not}` </evaluation>
<groundtruth> `{the ground truth answer}` </groundtruth>

Only return the trajectory summary of each step, e.g.,
1. what happened in the first step and the core outcomes
2. what happened in the second step and the core outcomes
3. ...

Figure 7: Prompt for summarizing each trajectory during Training-free GRPO in math tasks.

B Case Study

See how the agent performs on MATH problems with versus without experience, as shown in Figure 10 and Figure 11.

C The Use Of Large Language Models

We clarify that no LLMs were employed in the writing or polishing of this paper. All content presented herein is the result of original research and critical evaluation by the authors.

An agent system is provided with a set of experiences and has tried to solve the problem multiple times with both successful and wrong solutions. Review these problem-solving attempt and extract generalizable experiences. Follow these steps:

1. Trajectory Analysis:
   – For successful steps: Identify key correct decisions and insights
   – For errors: Pinpoint where and why the reasoning went wrong
   – Note any important patterns or strategies used/missed
   – Review why some trajectories fail? Is there any existing experiences are missed, or experiences do not provide enough guidance?

2. Update Existing Experiences
   – Some trajectories may be correct and others may be wrong, you should ensure there are experiences can help to run correctly
   – You have three options: [modify, add, delete]
      ∗ modify: You can modify current experiences to make it helpful
      ∗ add: You can introduce new experiences to improve future performance
      ∗ delete: You can delete existing experiences
   – You can update at most {`max number of operations`} clear, generalizable lessons for this case
   – Before updating each experience, you need to:
      ∗ Specify when it would be most relevant
      ∗ List key problem features that make this experience applicable
      ∗ Identify similar problem patterns where this advice applies

3. Requirements for each experience that is modified or added.
   – Begin with general background with several words in the experience
   – Focus on strategic thinking patterns, not specific calculations
   – Emphasize decision points that could apply to similar problems

Please provide reasoning in details under the guidance of the above 3 steps. After the step-by-step reasoning, you will finish by returning in this JSON format as follows:

```json
[
  {
    "option": "modify",
    "experience": "the modified experience",
    "modified_from": "G17" # specify the ID of experience that is modified
  },
  {
    "option": "add",
    "experience": "the added experience",
  },
  {
    "option": "delete",
    "delete_id": "the deleted experience ID",
  }, ...
]
```

Note that your updated experiences may not need to cover all the options. You can only use one type of updates or choose to remain all experiences unchanged.

<problem> {`problem`} </problem>
<trajectories> {G `trajectories in the same group`} </trajectories>
<groundtruth> {`answer`} </groundtruth>
<experience> {`experiences`} </experience>

Figure 8: Prompt for group advantage computation based on group rollouts during Training-free GRPO in math tasks.

An agent system is provided with a set of experiences and has tried to solve the problem multiple times. From the reflections, some suggestions on the existing experiences have been posed. Your task is to collect and think for the final experience revision plan. Each final experience must satisfy the following requirements
1. It must be clear, generalizable lessons for this case, with no more than 32 words
2. Begin with general background with several words in the experience
3. Focus on strategic thinking patterns, not specific calculations
4. Emphasize decision points that could apply to similar problems
5. Avoid repeating saying similar experience in multiple different experiences

<experience> {`experiences`} </experience>
<suggested_updates> {`group advantage`} </suggested_updates>

Please provide reasoning in each of the suggestions, and think for how to update existing experiences
You have two update options: [modify, merge]
- modify: You can modify current experiences to make it helpful - merge: You can merge some similar experiences into a more general forms to reduce duplication
After generating the step-by-step reasoning, you need to give the final experience revision details by returning in this JSON format as follows:

```json
[
    {
        "option": "modify",
        "experience": "the modified experience",
        "modified_from": "G17" # specify the ID of experience that is modified
    },
    {
        "option": "merge",
        "experience": "the merged experience",
        "merged_from": ["C1", "C3", "S4", ...] # specify the str IDs of experiences that is merged from, at least 2 IDs are needed
    },
    {
        "option": "delete",
        "delete_id": "the deleted experience ID",
    }, ...
]
```

Note that your updated experiences may not need to cover all the options. You can only use one type of updates or choose to remain all experiences unchanged.

<problem> {`problem`} </problem>
<trajectories> {G `trajectories in the same group`} </trajectories>
<groundtruth> {`answer`} </groundtruth>

Figure 9: Prompt for optimizating experiential knowledge $\mathcal{E}$ based on group advantages in the same batch during Training-free GRPO on math tasks.

Tool-integrated Math Reasoning (no experiences)

Question
Rectangles $ABCD$ and $EFGH$ are drawn such that D, E, C, F are collinear. Also, A, D, H, G all lie on a circle. If $BC = 16$,$AB = 107$,$FG = 17$, and $EF = 184$, what is the length of CE?

Trajectory
[Agent] I'll place point D at the origin (0,0) and assume C is at (107,0) and A at (0,16) based on rectangle dimensions.
[Tool] Execute code to set up coordinate system.
[Agent] Now let me calculate the determinant condition for points A, D, G, H to be concyclic using a 4x4 matrix.
[Tool] Execute code for determinant calculation.
[Agent] I get $CE \approx 106.91$., but this doesn't seem right geometrically. Let me try different point orders.
[Tool] Eexecute code for various orders ($D - C - E - F$, $E - D - C - F$). [Agent] Get negative CE or values outside valid range, but not validate solutions.
[Tool] Execute code with errors.
[Agent] Final answer: $CE \approx 106.91$. (incorrect)

Figure 10: Example trajectory without experience in Math Reasoning

Tool-integrated Math Reasoning (with experiences)

Question
Rectangles $ABCD$ and $EFGH$ are drawn such that D, E, C, F are collinear. Also, A, D, H, G all lie on a circle. If $BC = 16$,$AB = 107$,$FG = 17$, and $EF = 184$, what is the length of CE?

Experiences
[1] When solving geometry problems with intersections, validate solutions lie within bounded regions or segments, not on extensions, to avoid extraneous answers.
...
[29] When constructing points on line extensions, carefully interpret directional phrases like 'beyond B'.
...
[37] In geometry problems with points on sides of a triangle and given segment lengths, first determine all three side lengths by summing the appropriate segments. For example, if points on AB, AC, BC are given with AM, MB, AN, NC, BP, PC, then AB = AM + MB, AC = AN + NC, BC = BP + PC. This ensures the triangle is consistently defined before coordinate placement or area calculations.
[38] n geometry problems with multiple intersecting lines and given angles that don't sum to 180°, look for the relationship where the sum of intersection angles equals the sum of vertex angles. This pattern often appears when lines cross and form various angles, particularly when right angle markers indicate vertical/horizontal references. Focus on geometric relationships rather than coordinate calculations, and verify that the given angles are measured from the indicated reference directions (often vertical or horizontal).
...

Trajectory
[Agent] I'll place point D at the origin $(0, 0)$ and assume points D, E, C, F are collinear in order $D - E - C - F$. (Experience 29: Correct interpretation of collinear order)
[Tool] Execute code to set up coordinate system with $D(0, 0)$, $E(107 - x, 0)$, $C(107, 0)$, $F(291 - x, 0)$.
[Agent] For rectangles: $A(0, 16)$, $B(107, 16)$, $H(107 - x, -17)$, $G(291 - x, -17)$. (Experience 37: Determine all side lengths from given segments)
[Agent] Use circle equation: $x^2 + y^2 + Dx + Ey + F = 0$. Substitute $A(0, 16)$, $D(0, 0)$, $H(107 - x, -17)$ to find $F = 0$, $E = -16$, and derive equation.
[Agent] Ensure $G(291 - x, -17)$ lies on same circle, leading to quadratic equation: $x^2 - 398x + 30576 = 0$. (Experience 38: Focus on geometric relationships)
[Tool] Execute code to solve quadratic: solutions $x = 294$ or $x = 104$.
[Agent] Select $x = 104$ as valid solution.
[Tool] Execute code to verify concyclicity and dimensions.
[Agent] Final answer: $CE = 104$. (correct)

Figure 11: Example trajectory with experiences in Math Reasoning