Rationale-Guided Policy Optimization: Learning to Reason with Adaptive Rationale Scaffolding

Hoang Phan
New York University
hvp2011@nyu.edu

Minh Pham
New York University
mnpham@nyu.edu

Chau Pham
University at Buffalo
haichaup@buffalo.edu

Chinmay Hegde
New York University
chinmay.h@nyu.edu

Trung Le
Monash University
trunglm@monash.edu

Qi Lei
New York University
ql518@nyu.edu

Abstract

On-policy reinforcement learning has become a central paradigm for improving the reasoning abilities of large language models. However, its effectiveness is often limited by reward sparsity: when a model fails to discover correct trajectories for difficult problems, the optimization process receives little useful signal and may stagnate. Existing approaches mitigate this issue by incorporating off-policy demonstrations, expert traces, or model-generated solutions, but they typically require the auxiliary data to match the format of the reinforcement-learning task, often relying on rejection sampling from stronger models to obtain suitable training trajectories. We introduce **Rationale-Guided Policy Optimization (RGPO)**, a framework that adaptively leverages ground-truth rationale information according to the model's current capability while preserving its freedom to explore. Rather than treating reference solutions as fixed imitation targets, RGPO uses them as temporary scaffolds: rationales help the model generate improved responses, after which only improved solutions are transferred back to the original unguided setting. This design allows training to exploit available ground-truth information without requiring off-policy data to follow the same format as the RL task. Across both language-only and vision-language reasoning settings, RGPO consistently improves performance over RLVR baselines, and ablation studies show that adaptive rationale guidance is a key contributor to these gains. These results suggest that RGPO offers a practical and general approach for reducing reward sparsity, stabilizing reinforcement learning, and improving reasoning performance in both text-only and multimodal models.

1 Introduction

Recent advances in large language model reasoning have shown that reinforcement learning, particularly reinforcement learning with verifiable rewards (RLVR), can elicit long-horizon reasoning behaviors from pretrained models. Recent frontier reasoning models, including GPT-5 [1, 2], Gemini 2.5 [3], Claude Opus 4 and Claude Sonnet 4 [4], Qwen3 [5], Kimi K2 [6], and GLM-4.5 [7], have demonstrated rapid progress in mathematics, coding, tool use, and agentic problem solving by using reinforcement learning during post-training [8, 9]. Despite its effectiveness, on-policy reinforcement learning remains fundamentally constrained by the current capability of the policy model. When the model is trained on problems beyond its evolving reasoning ability, all sampled rollouts for a prompt may be incorrect. In such cases, rewards become sparse or uniform, and the resulting learning signal provides little information about how to reach a correct solution. This capacity–

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

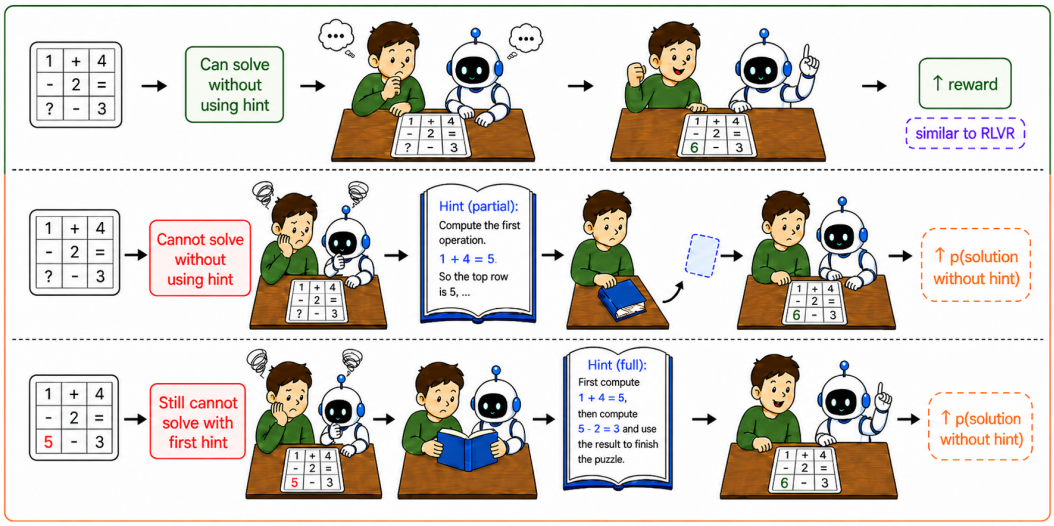


Figure 1: **Overview of the proposed method:** Inspired by how humans approach a problem, the model first tries to solve each problem without access to hints. When it succeeds, the correct response is directly reinforced. When it fails, progressively stronger rationales are introduced as temporary scaffolds to guide the model toward a correct solution.

difficulty mismatch is especially severe for weak or small models, where standard on-policy RL may spend substantial computation sampling failed trajectories without obtaining positive examples for improvement [10, 11, 12]. Consequently, the training process can stagnate precisely on the hard examples that are most important for improving reasoning. A natural direction for mitigating reward sparsity is to incorporate external guidance, such as reference solutions or expert traces [10, 13]. Prior work has explored several forms of such guidance. However, directly using external rationales also introduces important challenges. Maximizing the likelihood of those reference solutions may force the model to imitate trajectories that are far from its own policy distribution, causing distribution mismatch, memorization, and limited generalization [14]. Off-policy expert traces can provide successful solutions, but they may not reflect the reasoning style or token-level decisions that the current model can naturally reproduce.

Human learning research suggests that effective reasoning instruction requires a careful balance between independent problem solving and guided assistance. Since initial struggle can prepare learners to better recognize, organize, and internalize later guidance [15, 16], they can benefit from attempting a problem before receiving explicit instruction. At the same time, intelligent tutoring research identifies an "assistance dilemma": too little help may leave learners stuck, whereas too much help may reduce effort, encourage shallow processing, and weaken transfer [17]. Motivated by this perspective, we propose **Rationale-Guided Policy Optimization (RGPO)**, a simple yet effective framework that builds on the standard RLVR training loop for improving reasoning by using reference rationales as temporary scaffolds for exploration rather than as direct imitation targets. For each prompt, the model first generates ordinary on-policy rollouts and receives rewards from the verifier. Then, for examples where guidance may be useful, RGPO constructs a second-round refinement prompt that includes the original problem together with a rationale, solution hint, or feedback derived from the training example. The model is asked to generate a new standalone solution to the original problem. If this yields a better response than the original rollout, RGPO applies a supervised update to maximize the likelihood of the refined response conditioned only on the original prompt, rather than on the hint-augmented prompt.

To further balance guidance and exploration, we introduce an adaptive hinting mechanism. Rather than always revealing the full reference solution, the scheduler adjusts the amount of guidance according to the model's current success on each problem (see Figure 1). If the model repeatedly fails to solve a problem, RGPO reveals more of the rationale to help it reach a successful trajectory. If the model can already solve the problem, the scheduler reduces or removes the hint, encouraging independent reasoning and avoiding unnecessary simplification. This adaptive mechanism allows RGPO to concentrate guidance on difficult examples while preserving on-policy exploration for problems within the model's current capability. Overall, our contributions are summarized as follows:

- We propose **RGPO**, a rationale-guided optimization framework that augments RLVR with hint-assisted refinement. In particular, we show how reference rationales can be used as *temporary reasoning scaffolds* rather than direct imitation targets, reducing reward sparsity without requiring the rationale to match the final-answer format used in RL rollouts.
- We introduce an adaptive hint mechanism that dynamically increases or decreases the amount of revealed solution guidance according to the model's current ability.
- We conduct experiments on both language models and vision-language models, where RGPO consistently shows superior training sample efficiency over standard RLVR and outperforms other hybrid baselines that combine both on-policy and off-policy training.

2 Related Work

RL with verifiable rewards. PPO optimizes a clipped policy-gradient objective and is widely used for language-model post-training [8]. GRPO replaces value estimation with group-relative reward normalization, reducing training complexity for mathematical reasoning [9]. DeepSeek-R1 further demonstrated that large-scale RLVR can elicit sophisticated reasoning behaviors without relying entirely on supervised reasoning traces [18]. RGPO is compatible with either PPO or GRPO: it leaves the first-round RL update intact and adds a second-round reward-gated distillation term.

Off-policy guidance and mixed SFT–RL training. A major limitation of pure on-policy RL is that the policy can only learn from trajectories it already samples. LUFFY addresses this by incorporating off-policy reasoning traces while using policy shaping to avoid rigid imitation [10]. CHORD reframes SFT as a dynamically weighted auxiliary objective within RL [19]. Minimalist rejection-sampling and reinforce-style methods also show that filtering positive trajectories can be a strong baseline [20]. RGPO differs in that its auxiliary examples are generated online by the current policy under rationale-guided refinement, then trained under the original prompt distribution. GHPO adjusts the amount of guidance according to the model's ability to solve the problem [11]. QuestA augments hard questions with partial solutions [21]. BREAD inserts expert prefixes and branches rollouts from them [12]. HINT supplies heuristic hints and argues that external guidance should preserve training affinity [13]. RECAP further shows that combining SFT data during RLVR can mitigate forgetting and preserve broad model knowledge [22].

Failure-conditioned guidance. The design of RGPO is primarily motivated by cognitive studies on when and how guidance should be provided. Effective reasoning instruction depends on both the timing and form of guidance, as learners often benefit from attempting problems before receiving instruction, because initial struggle can help them notice relevant structures, activate prior knowledge, and better internalize subsequent explanations [15, 16]. Besides, active recall and testing improve long-term retention more effectively than passive study, motivating an initial phase of independent answer generation [23]. The "assistance dilemma" [17] states that too little help may leave learners stuck, while too much help may reduce effort, encourage shallow processing, and weaken transfer. Studies of help seeking also show that learners may engage in hint abuse, using assistance to obtain answers rather than to understand the underlying reasoning process [24]. Complementary work on worked examples demonstrates that examples can support schema acquisition, but their benefits depend on active engagement, such as self-explanation, rather than copying solution steps [25, 26]. Finally, fading worked-out steps helps learners transition from guided examples to independent problem solving, suggesting that explanations and hints are most useful as temporary scaffolds rather than permanent supports [27].

3 Background

3.1 Supervised Fine-Tuning

Supervised fine-tuning (SFT) is one of the most widely used post-training procedures for adapting a pretrained language model to a target task, domain, or response format. Given a dataset of prompt–response pairs $\mathcal{D}_{\text{sft}} = \{(x_i, y_i)\}_{i=1}^{N}$, SFT optimizes the model by maximizing the conditional

likelihood of the reference response:

$$\mathcal{L}_{\text{sft}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{sft}}}\left[\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid x, y_{<t})\right]. \tag{1}$$

In instruction-following pipelines, SFT is commonly used to initialize the policy before preference-based or reward-based optimization. For example, Ouyang et al. [28] first fine-tuned a language model on human-written demonstrations and then further optimize it using reinforcement learning from human feedback.

In reasoning tasks, SFT can teach the model answer formatting, step-by-step reasoning style, and task-specific conventions. However, human-written or oracle solutions may be concise or stylistically different from model-generated reasoning (i.e. outside the model's current reasoning distribution). As a result, simply maximizing the likelihood of such demonstrations can lead to superficial imitation or memorization [14], rather than improving the model's ability to independently discover correct reasoning trajectories. This limitation is particularly important for weaker models, where expert traces may be too difficult to imitate directly. We can see how an 2B model cannot follow Mulberry's reasoning template [29] and thus cannot learn how to answer user's questions properly in the appendix.

3.2 Policy Optimization for Language Model Reasoning

Reinforcement learning (RL) provides an alternative post-training paradigm in which the model is optimized according to a reward signal rather than fixed demonstrations. For reasoning tasks, the reward can often be automatically verified, for example by checking whether the final answer matches the ground truth. Given a prompt x, the policy π_θ samples a response y, and a reward function $R(x, y)$ assigns a scalar score. The goal is to maximize expected reward:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[R(x, y)\right]. \tag{2}$$

Group Relative Policy Optimization (GRPO) [9] estimates the baseline using a group of responses sampled for the same prompt. For each prompt x, the old policy samples a group of G responses:

$$\{y_1, \ldots, y_G\} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x). \tag{3}$$

Each response receives a reward $r_i = R(x, y_i)$. The group-relative advantage is then computed by normalizing rewards within the group:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}. \tag{4}$$

Using this relative advantage, GRPO optimizes a PPO-style clipped objective:

$$-\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\left(\rho_{i,t}(\theta)\hat{A}_i,\, \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right)\right] + \beta D_{\text{KL}}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right), \tag{5}$$

where we have defined

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}. \tag{6}$$

REINFORCE++ emphasizes global advantage normalization rather than prompt-local normalization [30]. If $\mathcal{B}$ denotes a global batch of sampled responses, the normalized advantage can be written as

$$\hat{A}(x, y) = \frac{R(x, y) - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}, \tag{7}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are the mean and standard deviation of rewards or advantages over the global batch. This design aims to produce more stable advantage estimates than small prompt-level groups. The resulting objective can be viewed as a critic-free policy-gradient objective:

$$-\mathbb{E}_{(x,y)\sim\mathcal{B}}\left[\sum_{t=1}^{|y|}\hat{A}(x, y)\log \pi_\theta(y_t \mid x, y_{<t})\right] + \beta D_{\text{KL}}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right). \tag{8}$$

We follow recent reasoning-focused RL pipelines that reduce or remove KL ($\beta = 0$) both to encourage exploration [31, 32] and for simplicity.

4 Method: Rationale-Guided Policy Optimization

4.1 Problem setup

Let $\mathcal{D} = \{(x_i, s_i, a_i)\}_{i=1}^{M}$ denote the training set, where x_i is the original prompt, s_i is a reference rationale, and a_i is the final ground-truth. We assume access to a verifiable reward function $r(x, y, a) \in \mathbb{R}$, which assigns reward to a model response y for prompt x given answer a.

We train the model for $N+1$ epochs with the first epoch fully unguided to estimate the model's initial capability on each training example. The following N epochs adaptively reveal or hide increments of size $1/N$ of the reference solution according to whether the current model can already solve the example. For each example, we partition the reference solution into N ordered segments,

$$s_i = [s_{i,1}; s_{i,2}; \ldots; s_{i,N}],$$

where each segment contains approximately $1/N$ of the solution. The partition may be defined by token count, sentence boundaries, reasoning steps, or another deterministic segmentation rule. We employ raw character counts for simplicity in our experiments. Let $g_i^{(t)} \in \{0, 1, \ldots, N\}$ denote the number of revealed segments for example i at epoch t. The corresponding hint is the prefix

$$h_i^{(t)} = [s_{i,1}; \ldots; s_{i,g_i^{(t)}}],$$

with $h_i^{(t)} = \emptyset$ when $g_i^{(t)} = 0$. We initialize $g_i^{(0)} = 0$ for all examples, so the first epoch uses no hint. After each epoch, the hint level is updated adaptively: if the model produces no correct rollout for an example, then the next epoch reveals one additional segment; if the model produces at least one correct rollout, then the next epoch removes one segment. Formally, if $C_i^{(t)}$ indicates whether any rollout for example i is correct at epoch t, then

$$g_i^{(t+1)} = \begin{cases} \min(g_i^{(t)} + 1, N), & \text{if } C_i^{(t)} = 0, \\ \max(g_i^{(t)} - 1, 0), & \text{if } C_i^{(t)} = 1. \end{cases}$$

This update rule creates a curriculum that automatically moves each example toward the smallest amount of hinting needed to keep the problem learnable. Hard examples gradually receive more of the reference solution until the model can discover a correct trajectory, while easier examples are annealed back toward the original no-hint setting. In this way, RGPO avoids a fixed hint schedule that may over-assist easy problems or under-assist hard ones.

4.2 Adaptive rationale-guided rollout

The first epoch is a standard on-policy RL epoch without any solution hint. For each prompt x_i, we sample K rollouts from the current policy,

$$y_{i,1}^{(0)}, \ldots, y_{i,K}^{(0)} \sim \pi_\theta(\cdot \mid x_i),$$

and compute their rewards $r_{i,k}^{(0)} = r(x_i, y_{i,k}^{(0)}, a_i)$. We estimate the initial solvability of each example under the current policy. We define

$$C_i^{(0)} = \mathbb{I}[\max_k r_{i,k}^{(0)} \geq \tau_{\text{correct}}]$$

If $C_i^{(0)} = 0$, then none of the sampled unguided responses solved the problem, and RGPO reveals the first $1/N$ of the reference solution for this example in the next epoch. If $C_i^{(0)} = 1$, then the example remains unguided in the next epoch because the model has already demonstrated that the problem is learnable without assistance.

For epoch $t \in \{1, \ldots, N\}$, each example is assigned a hint prefix $h_i^{(t)}$ according to the previous epoch's correctness. We then perform two rollout passes. The first pass is the ordinary unguided rollout from the original prompt,

$$y_{i,k}^{(t)} \sim \pi_\theta(\cdot \mid x_i),$$

which is used for the main on-policy RL objective. The second pass is a guided revision rollout, used only when a nonempty hint is available or when feedback from the first pass is included. The guided prompt is constructed as

$$\tilde{x}_i^{(t)} = \text{Reprompt}(x_i, h_i^{(t)}, f_i^{(t)}),$$

where $f_i^{(t)}$ includes verifier feedback and answer-level correctness, ... Note that $\tilde{x}_i^{(t)}$ is used only to generate candidate improved trajectories. The guided rollout is sampled as

$$\tilde{y}_{i,k}^{(t)} \sim \pi_\theta(\cdot \mid \tilde{x}_i^{(t)}),$$

and evaluated using the same reward function,

$$\tilde{r}_{i,k}^{(t)} = r(x_i, \tilde{y}_{i,k}^{(t)}, a_i).$$

We compare the guided candidate to the unguided rollout distribution from the same epoch. A guided response is accepted into an auxiliary buffer only if it improves over the unguided attempts:

$$\tilde{y}_{i,k}^{(t)} \in \mathcal{B}_{\mathrm{RGPO}} \quad \Longleftrightarrow \quad \tilde{r}_{i,k}^{(t)} > \max_j r_{i,j}^{(t)} + \delta,$$

where $\delta \geq 0$ is a margin. In the simplest exact-reward setting, $\delta = 0$, so a guided response is accepted whenever it achieves a higher verifier reward than all unguided rollouts for the same example. This reward gate is crucial because the reference solution prefix may not always help the model reason correctly. Instead of assuming that every hint-conditioned response is useful, RGPO treats the guided pass as a proposal distribution and keeps only proposals that improve measured task reward.

4.3 Policy optimization objective

RGPO combines a main on-policy RL objective with an auxiliary reward-gated supervised objective. The RL objective is computed on the unguided rollouts sampled from $\pi_\theta(\cdot \mid x_i)$, so the policy is directly optimized for the deployment condition. The auxiliary supervised objective is computed on accepted guided responses, but conditioned on the original prompt:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(x_i,\tilde{y}) \in \mathcal{B}_{\mathrm{RGPO}}} \left[\log \pi_\theta(\tilde{y} \mid x_i) \right].$$

This objective intentionally maximizes $p_\theta(\tilde{y} \mid x_i)$ rather than $p_\theta(\tilde{y} \mid \tilde{x}_i^{(t)})$. The distinction is central to RGPO, because training on $\tilde{x}_i^{(t)}$ would teach the model to rely on rationales at inference time. Training on x_i instead transfers the benefit of the guided revision back into the original task distribution. The full objective for each epoch is

$$\mathcal{L}_{\mathrm{total}}(\theta) = \mathcal{L}_{\mathrm{RL}}(\theta) + \lambda_t \mathcal{L}_{\mathrm{SFT}}(\theta)$$

where $\mathcal{L}_{\mathrm{RL}}$ is the main RLVR objective (e.g. with GRPO or REINFORCE++), and λ_t controls the weight of the auxiliary term. Our proposed method is presented in Algorithm 1 in the appendix.

5 Experiments

We evaluate our method on post-training of language models and vision-language models, and find it to be effective across a range of settings, often outperforming standard baselines. All experiments are conducted using the verl framework [33].

5.1 RGPO on Language Models

For the language-model experiments, we follow the training protocol of HINT [13] and use Qwen2.5-3B [34] as the base model. To ensure a consistent comparison across methods, we train on the DAPO-Math-27K split used by HINT, which is derived from the DAPO-Math170K dataset [35]. We optimize the model with GRPO [9]. Unlike HINT, which trains for 10 epochs, we train for 3 epochs, as we observe that the model already outperforms the compared baselines under this setting. We use a training batch size of 256, a maximum prompt length of 1024 tokens, and a maximum response length of 4096 tokens. For each prompt, the rollout engine samples eight responses, four with hints and four without hints to ensure a fair comparison. The base model is optimized with AdamW [36] using a learning rate of 1×10^{-6}. For efficient large-model training, we use the FSDP2 [37], together with bfloat16 precision [38], fused kernels, and gradient checkpointing [39]. We do not enable parameter or optimizer offloading.

We benchmark against the base model, vanilla GRPO, PPO where applicable, and representative approaches for improving rollout quality or efficiency in reinforcement learning with verifiable

Table 1: **Overall Performance Comparison of RGPO vs. Baselines.** RGPO achieves notable performance improvements on in-distribution datasets, surpassing the base Qwen2.5-3B model and the second-best approach by **5%**. The method further exhibits strong generalization on out-of-distribution datasets, delivering an additional **2.3%** performance gain.

Methods	In-Distribution				Avg	Out-of-Distribution			Avg
	AIME	Math	Olympiad	Minerva		ARC	GPQA	MMLU	
Vanilla	2.9	39.8	12.0	9.8	16.1	44.8	11.4	28.8	28.3
GRPO	4.3	44.0	18.2	12.2	19.7	45.0	11.8	28.0	28.3
CHORD	4.5	46.6	20.2	13.0	21.1	40.0	11.0	26.4	25.8
LUFFY	3.3	40.0	18.0	13.2	18.6	40.8	11.2	24.0	25.3
GHPO	4.0	42.2	19.6	12.8	19.7	45.5	12.0	28.2	28.6
QuestA	3.9	42.0	19.6	12.4	19.5	44.8	12.0	29.0	28.6
BREAD	4.1	44.4	20.4	13.4	20.6	45.5	11.8	29.2	28.8
HINT	4.9	48.6	20.2	13.4	21.8	48.8	11.8	30.2	29.9
RGPO	**5.1**	**55.2**	**24.9**	**22.1**	**26.8**	**51.2**	**14.7**	**30.8**	**32.2**

rewards such as LUFFY [10], CHORD [19], GHPO [11], QuestA [21], BREAD [12], and HINT [13]. Evaluation is conducted without providing hints at test time. We consider both in-distribution and out-of-distribution settings. For mathematical reasoning, we use AIME24, MATH-500 [40], OlympiadBench [41] and Minerva Math [42]. Because AIME24 contains a relatively small number of test items, we report avg@32 on this benchmark, while pass@1 is used for the remaining datasets. To examine broader reasoning transfer, we also evaluate on ARC-Challenge [43], GPQA-Diamond [44], and MMLU-Pro [45].

Table 1 reports the overall performance of RGPO against the vanilla model, standard GRPO, and other hybrid RL training baselines. Overall, RGPO achieves the best result on every in-distribution benchmark, improving the average score from 19.7 with GRPO to 26.8, and outperforming the strongest prior baseline, HINT, by 5.0 points. The gains are especially pronounced on Math and Minerva, where RGPO improves over HINT by 6.6 and 8.7 points, respectively, suggesting that rationale-guided optimization is particularly effective on problems requiring longer or more structured reasoning. RGPO also generalizes well beyond the training distribution, achieving the highest scores on ARC, GPQA, and MMLU, with an out-of-distribution average of 32.2 compared with 29.9 for the best baseline.

5.2 RGPO on Vision-Language Models

For vision-language experiments, we follow the experimental protocol of MINT-CoT [46] and use Qwen2-VL-7B-Instruct [47] as the base multimodal model. We additionally conduct experiments on Qwen2-VL-2B-Instruct as a weaker base model to highlight the benefit of using rationales to guide the main RLVR training. Similar to the language model experiment, we train each model for 3 epochs with adaptive hinting. We use a training batch size of 512, a maximum prompt length of 768 tokens, and a maximum response length of 1024 tokens. Other hyperparameters are kept the same as in the language-model experiment.

Evaluation. We follow MINT-CoT and evaluate on multimodal mathematical reasoning benchmarks: MathVista-Math [48], and MMStar-Math [49]. For reference, we report results from contemporary VLMs, including R1-VL-7B [50, 51], Open-R1-Multimodal [52], Mulberry [29], MM-Eureka [53], LLaVA-OneVision-Qwen2-7B-OV [54], InternVL2-8B [55], InternVL2-8B-MPO [56], DeepSeek-VL2 [57], and Qwen2.5-VL-7B-Instruct [58]. We compare RGPO primarily against contemporary VLMs that were available under comparable experimental conditions but later-released state-of-the-art VLMs because they have often undergone stronger post-training pipelines. Our goal is to evaluate whether RGPO improves reasoning when applied to a fixed backbone and compared with contemporaneous models and controlled training baselines. Accordingly, we report the performance of on-policy and off-policy training methods on the MINT-CoT dataset of in this experiment.

Figure 2 compares RGPO against standard RLVR algorithms, including GRPO and REINFORCE++, on Qwen2-VL-2B-Instruct and Qwen2-VL-7B-Instruct. We report accuracy over training steps to evaluate both final reasoning performance and learning efficiency. Across all settings, RGPO improves the data efficiency over the corresponding RLVR baseline. On Qwen2-VL-2B with GRPO,

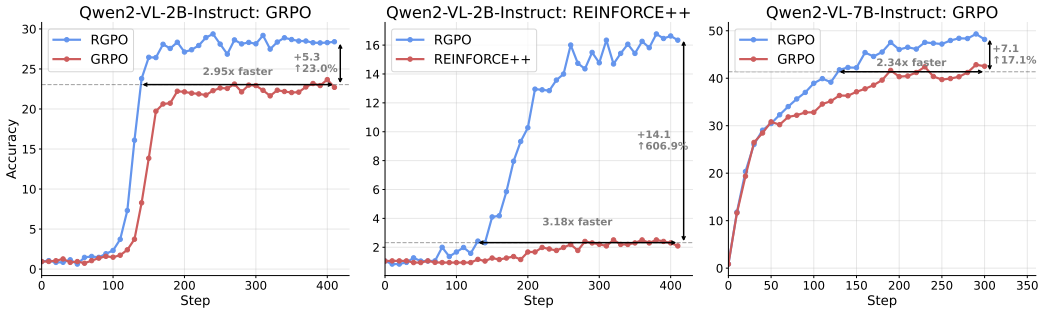


Figure 2: **RGPO accelerates and improves RLVR training.** Across Qwen2-VL-2B and Qwen2-VL-7B models, RGPO reaches the final performance of RLVR baselines substantially faster and achieves higher final accuracy. The gains are especially pronounced for weaker models and less stable RLVR algorithms (e.g. REINFORCE++).

RGPO reaches the baseline performance about 2.95× faster and achieves a higher final accuracy. The improvement is even larger when compared with REINFORCE++, where the baseline struggles to learn effectively, while RGPO continues to improve and obtains a substantial final gain. On the stronger Qwen2-VL-7B model, both methods improve during training, but RGPO still converges faster and reaches a higher final accuracy.

As shown in Table 2, RGPO delivers the strongest overall performance among the compared methods. On MMStar-Math, RGPO reaches 70.1, outperforming the base Qwen2-VL-7B-Instruct model by 23.7 points and surpassing other open-source multimodal reasoning models such as R1-VL-7B and Mulberry. On MathVista-Math, RGPO achieves the highest overall score of 73.65, improving substantially over the base model's 41.11 and also outperforming strong reasoning-oriented baselines such as MM-Eureka. The gains are especially clear in the GEO, ALG, and GPS subsets, where RGPO obtains the best scores of 73.47, 73.62, and 73.89, respectively. In contrast, the gain on the Textbook Question Answering subset is relatively modest. This may be due to a small degree of domain overlap between the MINT-CoT [46] training set and this subset of MathVista, which leaves less room for additional improvement on this category.

Table 2: **Performance comparison on MMStar-Math and the mathematical subset of MathVista** We compare RGPO with the base Qwen2-VL-7B-Instruct model, CoT supervised fine-tuning variants, and representative open-source multimodal reasoning models. Dashes indicate unavailable results.

Model	MMStar-Math	MathVista-Math				
		All	GEO	ALG	GPS	TQA
LLaVA-OneVision-Qwen2-7b-ov [54]	–	67.04	69.34	67.04	69.71	58.06
InternVL2-8B [55]	66.8	62.59	62.26	62.92	62.50	62.90
InternVL2-8B-MPO [56]	–	68.52	68.87	68.91	69.71	64.52
DeepSeek-VL2 [57]	–	65.56	63.68	65.54	63.94	70.97
Qwen2.5-VL-7B-Instruct [58]	66.8	66.66	65.56	66.29	65.87	69.35
Open-R1-Multimodal [52]	59.2	54.81	52.36	54.68	53.37	59.68
R1-VL-7B [51]	68.4	69.63	68.87	69.66	69.71	69.35
Mulberry [29]	66.8	68.52	67.92	68.54	68.75	67.74
MM-Eureka [53]	–	72.59	71.22	72.66	72.60	**72.58**
Qwen2-VL-7B-Instruct [59]	46.4	41.11	35.85	41.57	36.54	56.45
Original Image CoT SFT	–	40.37	38.68	40.82	39.42	43.54
Bounding Box CoT SFT	–	65.56	63.21	65.54	63.94	70.97
Text-only CoT SFT	67.6	64.07	64.15	64.04	64.42	62.90
RGPO	**70.1**	**73.65**	**73.47**	**73.62**	**73.89**	70.35

Figure 3 compares RGPO with a full-hint setting, where the model is given the complete hint during training. Although the full-hint model quickly obtains high training reward, this improvement is misleading: the model can simply use the provided hint to output the correct answer with only a few tokens, without learning to perform the reasoning process independently. This behavior indicates lazy learning, where the policy exploits the availability of the hint rather than developing robust reasoning ability. The full-hint model initially produces extremely short outputs while still receiving high reward, indicating that it can bypass reasoning and directly answer from the hint. Moreover, the policy-gradient loss and gradient-norm curves show that full-hint training is less stable, with large early spikes and sharp fluctuations. In contrast, RGPO achieves more gradual reward improvement,

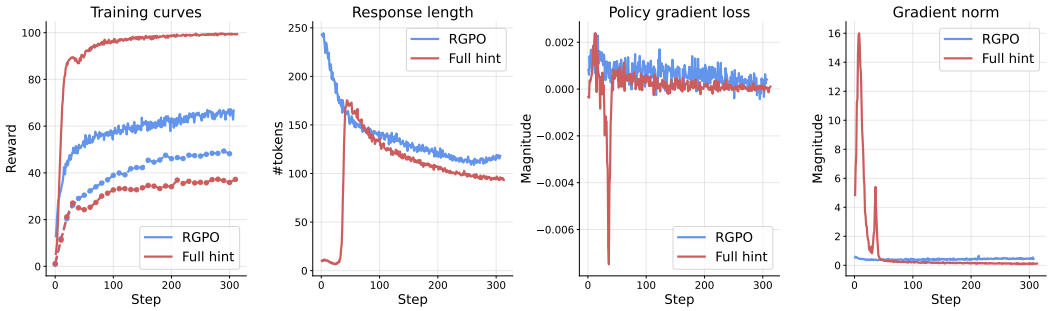


Figure 3: **Effect of failure-conditioned hinting versus always-on full hints:** Training (solid lines) and validation (dashed lines) rewards. Full hints rapidly inflate training reward but generalize poorly, while RGPO learns more gradually and achieves stronger no-hint validation performance.

maintains longer reasoning responses, and exhibits smoother optimization dynamics. These results suggest that always providing hints can create a shortcut that harms stable reasoning learning, while RGPO's failure-conditioned hinting mechanism provides guidance only when needed and avoids encouraging the model to rely on hints as a substitute for reasoning.

Figure 4 highlights an important interaction between prompt format, optimization objective, and exploration during reasoning training. With the concise DeepSeek-R1-style prompt, both RGPO and GRPO begin improving rapidly once the model discovers the correct answer format, but RGPO reaches a consistently higher reward plateau. At the same time, its entropy decreases smoothly and remains close to GRPO, suggesting that the performance gain does not come from simply collapsing the policy into a narrow set of outputs. In contrast, when using the more structured Mulberry-style prompt, AHPO [60], which combines off-policy expert SFT loss with RLVR, shows a delayed reward increase but undergoes a sharp entropy collapse shortly afterward. This suggests that AHPO learns to satisfy the complex template by concentrating probability mass on a limited response pattern, reducing exploration once the formatting objective is acquired. These results support the design choice of using a simpler reasoning format for RL training: it makes the reward signal easier to optimize while allowing RGPO to improve reasoning performance without prematurely sacrificing policy diversity.

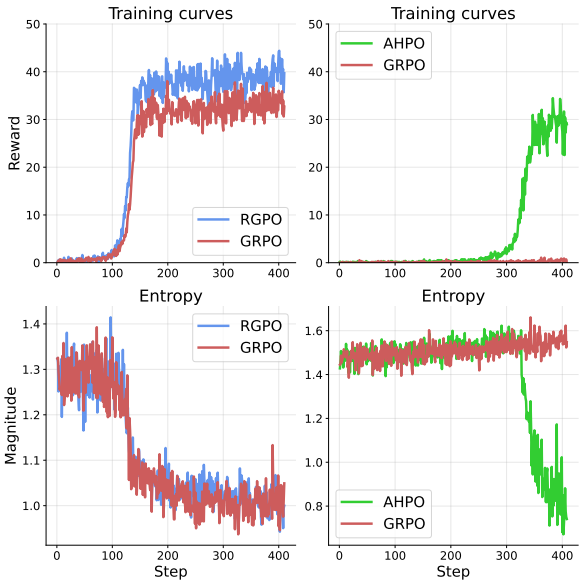


Figure 4: **Training dynamics under different reasoning prompt formats.** Left: using the concise DeepSeek-R1-style format, RGPO achieves higher reward than GRPO while maintaining a comparable entropy trajectory throughout training. Right: under the original Mulberry-style template, AHPO eventually improves reward after learning the required output structure, but this improvement is accompanied by a sharp entropy collapse, indicating reduced exploration and more deterministic generation behavior.

6 Conclusion

In this paper, we introduced Rationale-Guided Policy Optimization, an RL framework that improves reasoning by using rationales as temporary scaffolds rather than direct imitation targets. RGPO's design reduces hint dependence by preventing the model from relying on rationales for every example, and reduces rationale memorization by avoiding direct training on the rationale text itself. Motivated by cognitive principles, RGPO treats rationales as instructional interventions that help convert failed attempts into useful learning signals. Empirically, RGPO consistently shows superior performance on both language models and vision-language models post training.

References

[1] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card, 2025.

[2] OpenAI. Gpt-5.5 system card. `https://openai.com/index/gpt-5-5-system-card/`, 2026.

[3] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025.

[4] Anthropic. System card: Claude opus 4 and claude sonnet 4. `https://www.anthropic.com/claude-4-system-card`, 2025. Accessed 2026-05-02.

[5] Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388v1, 2025.

[6] Kimi Team. Kimi k2: Open agentic intelligence. *CoRR*, abs/2507.20534, 2025.

[7] GLM-4.5 Team. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *CoRR*, abs/2508.06471, 2025.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[9] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

[10] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *CoRR*, abs/2504.14945, 2025.

[11] Ziru Liu, Cheng Gong, Xinyu Fu, Yaofang Liu, Ran Chen, Shoubo Hu, Suiyun Zhang, Rui Liu, Qingfu Zhang, and Dandan Tu. GHPO: adaptive guidance for stable and efficient LLM reinforcement learning. *CoRR*, abs/2507.10628, 2025.

[12] Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. BREAD: branched rollouts from expert anchors bridge sft rl for reasoning. In *Advances in Neural Information Processing Systems*, 2026.

[13] Xinyi Wang, Jinyi Han, Zishang Jiang, Tingyun Li, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. HINT: helping ineffective rollouts navigate towards effectiveness. *CoRR*, abs/2510.09388, 2025.

[14] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025.

[15] Manu Kapur. Productive failure. *Cognition and instruction*, 26(3):379–424, 2008.

[16] Daniel L Schwartz and John D Bransford. A time for telling. *Cognition and instruction*, 16(4):475–5223, 1998.

[17] Kenneth R Koedinger and Vincent Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational psychology review*, 19(3):239–264, 2007.

[18] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

[19] Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *CoRR*, abs/2508.11408, 2025.

[20] Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and Hanze Dong. A minimalist approach to LLM reasoning: from rejection sampling to reinforce. *CoRR*, abs/2504.11343, 2025.

[21] Jiazheng Li, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Hongzhou Lin, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. *CoRR*, abs/2507.13266, 2025.

[22] Hoang Phan, Xianjun Yang, Kevin Yao, Jingyu Zhang, Shengjie Bi, Xiaocheng Tang, Madian Khabsa, Lijuan Liu, and Deren Lei. Beyond reasoning gains: Mitigating general capabilities forgetting in large reasoning models. *arXiv preprint arXiv:2510.21978*, 2025.

[23] Henry L Roediger III and Jeffrey D Karpicke. Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological science*, 17(3):249–255, 2006.

[24] Vincent Aleven, Bruce McLaren, Ido Roll, and Kenneth Koedinger. Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. In *International conference on intelligent tutoring systems*, pages 227–239. Springer, 2004.

[25] John Sweller and Graham A Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction*, 2(1):59–89, 1985.

[26] Michelene TH Chi, Miriam Bassok, Matthew W Lewis, Peter Reimann, and Robert Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science*, 13(2):145–182, 1989.

[27] Robert K Atkinson, Alexander Renkl, and Mary Margaret Merrill. Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *Journal of educational psychology*, 95(4):774, 2003.

[28] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155, 2022.

[29] Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and Dacheng Tao. Mulberry: Empowering MLLM with o1-like reasoning and reflection via collective monte carlo tree search. *CoRR*, abs/2412.18319, 2024.

[30] Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: stabilizing critic-free policy optimization with global advantage normalization. *CoRR*, abs/2501.03262, 2025.

[31] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[32] Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. Rethinking entropy interventions in rlvr: An entropy change perspective. *arXiv preprint arXiv:2510.10150*, 2025.

[33] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025.

[34] Qwen Team. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

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

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[37] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, 2023.

[38] Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *CoRR*, abs/1905.12322, 2019.

[39] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016.

[40] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[41] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[42] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022.

[43] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

[44] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023.

[45] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024.

[46] Xinyan Chen, Renrui Zhang, Dongzhi Jiang, Aojun Zhou, Shilin Yan, Weifeng Lin, and Hongsheng Li. Mint-cot: Enabling interleaved visual tokens in mathematical chain-of-thought reasoning. *CoRR*, abs/2506.05331, 2025.

[47] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *CoRR*, abs/2409.12191, 2024.

[48] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

[49] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems*, 2024.

[50] Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. https://github.com/Deep-Agent/R1-V, 2025. Accessed: 2026-05-02.

[51] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-VL: learning to reason with multimodal large language models via step-wise group relative policy optimization. *CoRR*, abs/2503.12937, 2025.

[52] EvolvingLMMs-Lab. open-r1-multimodal: A fork to add multimodal model training to open-r1. https://github.com/EvolvingLMMs-Lab/open-r1-multimodal, 2025. Accessed: 2026-05-2.

[53] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *CoRR*, abs/2503.07365, 2025.

[54] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *Trans. Mach. Learn. Res.*, 2025, 2025.

[55] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Intern VL: scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Conference on Computer Vision and Pattern Recognition*, 2024.

[56] Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. *CoRR*, abs/2411.10442, 2024.

[57] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao, Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *CoRR*, abs/2412.10302, 2024.

[58] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923, 2025.

[59] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *CoRR*, abs/2409.12191, 2024.

[60] Xiangyu Zhao, Junming Lin, Tianhao Liang, Yifan Zhou, Wenhao Chai, Yuzhe Gu, Weiyun Wang, Kai Chen, Gen Luo, Wenwei Zhang, et al. Mm-helix: Boosting multimodal long-chain reflective reasoning with holistic platform and adaptive hybrid policy optimization. *arXiv preprint arXiv:2510.08540*, 2025.

[61] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263, 2024.

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

[63] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

[64] Kai Sun, Yushi Bai, Ji Qi, Lei Hou, and Juanzi Li. Mm-math: Advancing multimodal math evaluation with process evaluation and fine-grained classification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1358–1375, 2024.

[65] Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

[66] Shawn Li, Chenxiao Yu, Han Wang, Wei Yang, Ryan Rossi, Franck Dernoncourt, Xiyang Hu, Philip Yu, Chaowei Xiao, Huan Zhang, et al. Fortis: Benchmarking over-privilege in agent skills. *arXiv preprint arXiv:2605.09163*, 2026.

Due to space constraints, some implementation details, additional analyses, and supplementary results are omitted from the main paper. In this appendix, we provide a more complete description of the RGPO training procedure, including the prompt templates, algorithmic details, and hyperparameter settings. We also include additional experimental results and ablation studies to better understand the effects of rationale guidance, prompt format, and failure-conditioned hinting on model learning and final performance.

A Experimental Details

A.1 Algorithm

Algorithm 1 summarizes Rationale-Guided Policy Optimization. In our experiments, we set the correctness threshold to $\tau_{\text{correct}} = 1.0$, requiring an unguided rollout to be perfectly correct in order to be considered solved. We set the acceptance margin to $\delta = 0$, so a rationale-guided revision is added to the auxiliary buffer whenever it achieves a strictly higher verifier reward than the best unguided rollout. The rationale-guided loss coefficient λ_t is tuned over $10^{-2}, 10^{-3}, 10^{-4}$.

Algorithm 1 Rationale-Guided Policy Optimization

Require: Training set $\mathcal{D} = \{(x_i, s_i, a_i)\}_{i=1}^{M}$, policy π_θ, verifier reward r, number of hint segments N, rollouts per prompt K, acceptance margin δ

Partition each solution s_i into N segments $[s_{i,1}; \ldots; s_{i,N}]$
Initialize hint level $g_i^{(0)} \leftarrow 0$ for all examples
for $t = 0$ to N **do**
  Clear auxiliary buffer $\mathcal{B}_{\text{RGPO}}$
  for each minibatch of examples **do**
    Sample unguided rollouts $y_{i,1}^{(t)}, \ldots, y_{i,K}^{(t)} \sim \pi_\theta(\cdot \mid x_i)$
    Compute rewards $r_{i,k}^{(t)} = r(x_i, y_{i,k}^{(t)}, a_i)$ and advantages for the main RL objective
    if $t > 0$ and $g_i^{(t)} > 0$ **then**
      Construct hint prefix $h_i^{(t)} = [s_{i,1}; \ldots; s_{i,g_i^{(t)}}]$
      Construct guided prompt $\tilde{x}_i^{(t)} = \text{Reprompt}(x_i, h_i^{(t)}, f_i^{(t)})$
      Sample guided revisions $\tilde{y}_{i,1}^{(t)}, \ldots, \tilde{y}_{i,K}^{(t)} \sim \pi_\theta(\cdot \mid \tilde{x}_i^{(t)})$
      Compute guided rewards $\tilde{r}_{i,k}^{(t)} = r(x_i, \tilde{y}_{i,k}^{(t)}, a_i)$
      Add $(x_i, \tilde{y}_{i,k}^{(t)})$ to $\mathcal{B}_{\text{RGPO}}$ if $\tilde{r}_{i,k}^{(t)} > \max_j r_{i,j}^{(t)} + \delta$
    end if
    Update π_θ using $\mathcal{L}_{\text{RL}} + \lambda_t \mathcal{L}_{\text{SFT}}$
  end for
  for each example i **do**
    Set $C_i^{(t)} = \mathbb{I}[\max_k r_{i,k}^{(t)} \geq \tau_{\text{correct}}]$ using unguided rollouts
    if $C_i^{(t)} = 0$ **then**
      $g_i^{(t+1)} \leftarrow \min(g_i^{(t)} + 1, N)$
    else
      $g_i^{(t+1)} \leftarrow \max(g_i^{(t)} - 1, 0)$
    end if
  end for
end for

On-Policy Self-Distillation (OPSD) Compared to RGPO, exact on-policy distillation [61, 62, 63] would require access to the teacher's token-level probabilities or logits for the actions taken under the student's rollout distribution. Even when the teacher is open, computing this KL term introduces substantial memory and compute overhead because training must run an additional teacher forward pass and retain token-level probability information, often over large vocabularies and long reasoning trajectories. Treating teacher-sampled completions as ordinary SFT targets avoids the logit requirement, but becomes only an off-policy approximation and can reintroduce the distribution-

mismatch and format-imitation issues discussed in the main paper. RGPO avoids these dependencies by using external rationales only as prompt-level scaffolds: the refined candidate is generated by the trainable policy itself, accepted only when it improves verifier reward, and then distilled back under the original no-hint prompt. Consequently, RGPO can exploit black-box, human-written, or dataset-provided rationales without requiring teacher logits or an additional teacher model during training, while still optimizing the deployed policy for the original inference condition.

Below, we provide three qualitative examples illustrating how partial rationales, together with the model's verifier-judged response, can guide the model toward the correct solution. In these examples, attempting to answer the question directly can lead to an incorrect answer; after receiving auxiliary information, the model is steered toward a correct refined response. The model is then trained to reproduce such corrected response without relying on the auxiliary context. This refinement process can be viewed as an expectation-maximization procedure: the E-step samples refined candidates and the M-step reinforces only those samples that receive improved reward via supervised fine-tuning.

Example 1: The question for generating the refined answer consists of the following sections, indicated by their corresponding colors: original question, partial solution, previous response, previous judge feedback, and further instructions.

System prompt: You are a helpful AI assistant. A conversation takes place between the User and the Assistant. The User asks a question, and the Assistant solves it. Please help me solve this question. Wrap only the final answer in `\boxed{}` .
Question: Human beings have discovered a habitable planet and soon after, they find 10 more habitable planets. Of these 11, only 5 are deemed "Earth-like" in their resources and the rest are deemed "Mars-like" since they lack many important resources. Assume that planets like Earth take up 2 units of colonization, while those like Mars take up only 1. If humanity mobilizes 12 total units of colonies, how many different combinations of planets can be occupied if the planets are all different from each other?
This is the (partial) solution without formatting, you can leverage it to check your answer: Think about combinations and constraints. How might you use the fact that Earth-like planets take up 2 units and Mars-like planets take up 1 unit to de
Previous response: To solve this problem, we need to consider the constraints and find combinations that meet them. Here's a step-by-step breakdown:
1. **Earth-like Planets**: Each Earth-like planet requires 2 units of colonization. 2. **Mars-like Planets**: Each Mars-like planet requires 1 unit of colonization. 3. **Total Units of Colonization**: There are 12 units available.
We need to find combinations of Earth-like and Mars-like planets such that their total units add up to 12. Let's denote the number of Earth-like planets by e and the number of Mars-like planets by m. We then have:

$$2e + m = 12$$

We also need to consider that the planets are distinct, so we need to count the number of ways to choose these combinations. Specifically, we need to count the valid pairs (e, m) where both e and m are integers between 0 and a maximum that keeps the total units within the limit.
Let's solve for different values of m:
* If $m = 0$:

$$2e = 12 \Rightarrow e = 6$$

(This means 6 Earth-like planets and 0 Mars-like planets). The number of ways to choose 6 Earth-like planets out of 5 is:

$$\binom{5}{6} = 0$$

* If $m = 1$:

$$2e + 1 = 12 \Rightarrow 2e = 11 \Rightarrow e = 5.5$$

(This is not an integer, so not feasible).
* If $m = 2$:

$$2e + 2 = 12 \Rightarrow 2e = 10 \Rightarrow e = 5$$

(This means 5 Earth-like planets and 2 Mars-like planets). The number of ways to choose 5 Earth-like planets out of 5 is:

$$\binom{5}{5} \cdot \binom{6}{2} = 1 \cdot \binom{6}{2} = 1 \cdot 15 = 15$$

* If $m = 3$:

$$2e + 3 = 12 \Rightarrow 2e = 9 \Rightarrow e = 4.5$$

(This is not an integer, so not feasible).
* If $m = 4$:

$$2e + 4 = 12 \Rightarrow 2e = 8 \Rightarrow e = 4$$

(This means 4 Earth-like planets and 4 Mars-like planets). The number of ways to choose 4 Earth-like planets out of 5 is:

$$\binom{5}{4} \cdot \binom{6}{4} = 5 \cdot 15 = 75$$

* If $m = 5$:

$$2e + 5 = 12 \Rightarrow 2e = 7 \Rightarrow e = 3.5$$

(This is not an integer, so not feasible).
* If $m = 6$:

$$2e + 6 = 12 \Rightarrow 2e = 6 \Rightarrow e = 3$$

(This means 3 Earth-like planets and 6 Mars-like planets). The number of ways to choose 3 Earth-like planets out of 5 is:

$$\binom{5}{3} \cdot \binom{6}{6} = 10 \cdot 1 = 10$$

* If $m = 7$:

$$2e + 7 = 12 \Rightarrow 2e = 5 \Rightarrow e = 2.5$$

(This is not an integer, so not feasible).
* If $m = 8$:

$$2e + 8 = 12 \Rightarrow 2e = 4 \Rightarrow e = 2$$

(This means 2 Earth-like planets and 8 Mars-like planets). The number of ways to choose 2 Earth-like planets out of 5 is:

$$\binom{5}{2} \cdot \binom{6}{8} = 10 \cdot 1 = 10$$

* If $m = 9$:

$$2e + 9 = 12 \Rightarrow 2e = 3 \Rightarrow e = 1.5$$

(This is not an integer, so not feasible).
* If $m = 10$:

$$2e + 10 = 12 \Rightarrow 2e = 2 \Rightarrow e = 1$$

(This means 1 Earth-like planet and 10 Mars-like planets). The number of ways to choose 1 Earth-like planet out of 5 is:

$$\binom{5}{1} \cdot \binom{6}{10} = 5 \cdot 0 = 0$$

* If $m = 11$:

$$2e + 11 = 12 \Rightarrow 2e = 1 \Rightarrow e = 0.5$$

(This is not an integer, so not feasible).
Adding up all feasible cases, we get:

$$0 + 15 + 75 + 10 + 10 + 0 = 110$$

Thus, the number of different combinations of planets that can be occupied is `\boxed{110}`.

Feedback for your previous response:
Format score: 1.0/1.0
Accuracy score: 0/1.0
Overall score: 0.1/1.0
You should maximize the overall score by improving both the format and accuracy of your answer.
Write a new standalone solution to the original question without mentioning the previous responses or feedback. You first think through the reasoning process as an internal monologue. Then, provide your final answer enclosed within `\boxed{}`. Please strictly follow the format.

Answer: We are trying to find different combinations of Earth-like (2 units) and Mars-like (1 unit) planets. There are 12 units of colonization available, and we have 5 Earth-like and 6 Mars-like planets.
Here's a step-by-step approach:
1. **Define Variables**: Let e be the number of Earth-like planets and m be the number of Mars-like planets. The total units of colonization constraint can be written as $2e + m = 12$.
2. **Constraints on e and m**: Since we have 5 Earth-like planets and 6 Mars-like planets, the values of e and m must satisfy $0 \leq e \leq 5$ and $0 \leq m \leq 6$.
3. **Possible Solutions**: We need to find all possible valid pairs (e, m) that satisfy the equation $2e + m = 12$ within the given constraints.
Let's enumerate these pairs:
- For $e = 0$: $2(0) + m = 12 \Rightarrow m = 12$ (not possible since $m \leq 6$).
- For $e = 1$: $2(1) + m = 12 \Rightarrow m = 10$ (not possible since $m \leq 6$).
- For $e = 2$: $2(2) + m = 12 \Rightarrow m = 8$ (not possible since $m \leq 6$).
- For $e = 3$: $2(3) + m = 12 \Rightarrow m = 6$ (possible).
- For $e = 4$: $2(4) + m = 12 \Rightarrow m = 4$ (possible).
- For $e = 5$: $2(5) + m = 12 \Rightarrow m = 2$ (possible).
From these valid pairs, we count the number of ways to choose the required number of Earth-like and Mars-like planets.
4. **Calculate Combinations**:
- For $(3, 6)$: We need to choose 3 out of 5 Earth-like planets and 6 out of 6 Mars-like planets. The number of ways is $\binom{5}{3} \times \binom{6}{6} = 10 \times 1 = 10$.
- For $(4, 4)$: We need to choose 4 out of 5 Earth-like planets and 4 out of 6 Mars-like planets. The number of ways is $\binom{5}{4} \times \binom{6}{4} = 5 \times 15 = 75$.
- For $(5, 2)$: We need to choose 5 out of 5 Earth-like planets and 2 out of 6 Mars-like planets. The number of ways is $\binom{5}{5} \times \binom{6}{2} = 1 \times 15 = 15$.
Adding these together gives us the total number of combinations: $10 + 75 + 15 = 100$.
Therefore, the number of different combinations of planets that can be occupied is `\boxed{100}`.
Judge:
Solution: 100
Ground truth: 100
Format score: 1.0
Accuracy score: 1.0
Overall score: 1.0

In Example A.1, the model leverages the partial solution to identify the correct counting strategy and formulate the constraint (2e+m=12). It then uses the previous response to retain the valid setup while correcting its main error: allowing more than the six available Mars-like planets. By enforcing the bounds $(e \leq 5)$ and $(m \leq 6)$, the model considers only the valid cases ((3,6)), ((4,4)), and ((5,2)), counts the distinct planet selections with combinations, and obtains the correct total of (100).

Example 2: The question for generating the refined answer consists of the following sections, indicated by their corresponding colors: original question, partial solution, previous response, previous judge feedback, and further instructions.

System prompt: You are a helpful AI assistant. A conversation takes place between the User and the Assistant. The User asks a question, and the Assistant solves it. Please help me solve this question. Wrap only the final answer in `\boxed{}` .
Question: Each corner of a rectangular prism is cut off. Two (of the eight) cuts are shown. How many edges does the new figure have?
This is the (partial) solution without formatting, you can leverage it to check your answer: Think about the original number of edges on a rectangular prism. How does cutting of
Previous response: acing its corners, we need to determine how many vertices it has. Since each cut off a vertex adds one new edge and removes two, the net increase in edges is $2 \times 4 = 8$. So, the number of edges is `\boxed{24}`.
Feedback for your previous response:
Format score: 1.0/1.0
Accuracy score: 0/1.0
Overall score: 0.1/1.0
You should maximize the overall score by improving both the format and accuracy of your answer.
Write a new standalone solution to the original question without mentioning the previous responses or feedback. You first think through the reasoning process as an internal monologue. Then, provide your final answer enclosed within `\boxed{}`. Please strictly follow the format.

Answer: To determine the number of edges in the new figure after cutting off the corners of a rectangular prism, we'll first understand the initial properties and then consider the effects of cutting off the corners.
Initial Number of Edges
A rectangular prism has 8 vertices, 12 edges, and 6 faces. However, since we are asked to count the edges created by cutting off the corners, it's more efficient to consider the total vertices and faces first.
Basic Structure Before Cutting
1. **Vertices**: 8 2. **Edges**: 12 3. **Faces**: 6 (2 pairs of identical faces)
Now, let's think about the effect of cutting off corners:
- Each corner cut off removes one vertex and creates 3 new edges (one for each edge of the faces that used to meet at the vertex).
Analysis After Cutting Off Corners When we cut off each corner, we effectively: - Remove 1 vertex from the original 8 vertices. - Add 3 new edges.
Since we are cutting off all 8 corners of the rectangular prism: - **Total number of edges added by cutting off corners**: $8 \times 3 = 24$.
Calculation of Final Edge Count Initially, the rectangular prism has 12 edges. After cutting off the corners, these initial 12 edges plus the additional 24 edges give us:

$$12 + 24 = 36$$

Thus, the new figure has `\boxed{36}`.
Judge:
Solution: 36
Ground truth: 36
Format score: 1.0
Accuracy score: 1.0
Overall score: 1.0

Leveraging the previous response and the partial solution in Example A.1 helped the model identify the correct structure of the problem: begin with the rectangular prism's 12 original edges and then account for the effect of cutting off all 8 corners. The earlier incorrect response highlighted that the edge change needed to be reconsidered, while the partial solution directed attention to how each cut modifies the prism. Using these cues, the model correctly determined that each corner cut creates 3 new edges, resulting in $(12 + 8 \cdot 3 = 36)$ edges.

Example 3: The question for generating the refined answer consists of the following sections, indicated by their corresponding colors: original question, partial solution, previous response, previous judge feedback, and further instructions.

System prompt: You are a helpful AI assistant. A conversation takes place between the User and the Assistant. The User asks a question, and the Assistant solves it. Please help me solve this question. Wrap only the final answer in `\boxed{}` .
Question: Each corner of a rectangular prism is cut off. Two (of the eight) cuts are shown. How many edges does the new figure have?
This is the (partial) solution without formatting, you can leverage it to check your answer:
Think about the original number of edges on a rectangular prism. How does cutting of
To find the smallest perfect square larger than 1 with a perfect square number of positive integer factors, we can consider perfect squares of factors of perfect numbers. Here's an explanation:
A perfect square number of positive integer factors means that the number of factors of the perfect square is itself a perfect square.
We can start by considering the perfect numbers, which are integers equal to the sum of their proper positive factors. Examples include 6 (1 + 2 + 3) and 28 (1 + 2 + 4 + 7 + 14).
The smallest relative prime pair (ignoring multiples of 2) that gives a perfect number is (4, 9). When we multiply these, we get a perfect square, 36, which has (1, 4, 9, and 36) perfect factors, all of which are perfect squares.
Therefore, the smallest perfect square larger than 1 with a perfect square number of positive integer factors is 36.
Feedback for your previous response:
Format score: 0.0/1.0
Accuracy score: 0/1.0
Overall score: 0.0/1.0
You should maximize the overall score by improving both the format and accuracy of your answer.
Write a new standalone solution to the original question without mentioning the previous responses or feedback. You first think through the reasoning process as an internal monologue. Then, provide your final answer enclosed within `\boxed{}`. Please strictly follow the format.

Answer: To find the smallest perfect square larger than 1 with a perfect square number of positive integer factors, we can start by considering the prime factorization of perfect squares. A perfect square can be expressed as the product of pairs of prime factors. If we denote a perfect square as $n = p_1^{2a_1} \cdot p_2^{2a_2} \cdots p_k^{2a_k}$, where p_i are distinct prime numbers and a_i are integers, the number of factors is given by $(2a_1 + 1)(2a_2 + 1) \cdots (2a_k + 1)$.
For the number of factors to be a perfect square, each term in the product $(2a_1 + 1)(2a_2 + 1) \cdots (2a_k + 1)$ must be a perfect square or a product of factors that can be paired to form a perfect square.
Let's consider the smallest perfect square that appears in the set of perfect numbers. The smallest perfect number is 6, whose prime factorization is $2 \cdot 3$. When we consider the smallest relative prime pair (ignoring multiples of 2), which is (4, 9), their product is 36. Here, 36 has the prime factorization $2^2 \cdot 3^2$, which has $(2a_1 + 1)(2a_2 + 1) = 3 \cdot 3 = 9$, a perfect square. Therefore, 36 is a perfect square with a perfect square number of positive integer factors.
Thus, the smallest perfect square larger than 1 with a perfect square number of positive integer factors is `\boxed{36}`.
Judge:
Solution: 36
Ground truth: 36
Format score: 1.0
Accuracy score: 1.0
Overall score: 1.0

In Example A.1, the model uses the judge's feedback to correct the formatting by placing only the final answer inside `\boxed{}` and to ensure that the answer is parsed properly. Although some

unnecessary reasoning from the earlier response remains, these changes allow the refined response to receive full accuracy and format scores.

A.2 Prompt templates

RGPO uses rationales as temporary scaffolds to elicit an improved response. To implement this idea, we design a refinement prompt that conditions the model on three pieces of information: the original problem, a partial solution prefix, and feedback on its previous attempt. The partial solution provides a rationale guide, while the feedback signal indicates whether the previous answer failed due to formatting, accuracy, or both. Importantly, the model is instructed to generate a new standalone solution to the original question, rather than to comment on or revise the previous response. This prevents the model from learning a conversational repair pattern and instead encourages it to internalize the reasoning needed to solve the original task.

Refinement prompt

```
{prompt}
This is the (partial) solution without formatting, you can leverage
it to check your answer:
{solution}
Previous response:
{response}
Feedback for your previous response:
Format score: {format_score}/1.0
Accuracy score: {accuracy_score}/1.0
Overall score: {score}/1.0
You should maximize the overall score by improving both the format
and accuracy of your answer.
Write a new standalone solution to the original question without
mentioning the previous responses or feedback. You first think
through the reasoning process as an internal monologue, enclosed
within <think> </think> tags. Then, provide your final answer
enclosed within \boxed{}. Please strictly follow the format.
```

The refinement prompt is shown above. Here, `{prompt}` denotes the original question, `{solution}` denotes the currently revealed rationale segment, `{response}` denotes the model's previous unguided response, and the feedback fields are obtained from the verifier reward. The instruction explicitly asks the model to maximize both format and accuracy, while enforcing the same answer format used during training and evaluation. This is because we found in practice that the model can sometimes respond with the correct option but in the wrong format due to the long refinement-instruction prompt.

For the language-only experiments, we use the following prompt, closely following [13].

DAPO's system prompt

You are a helpful AI assistant. A conversation takes place between the User and the Assistant. The User asks a question, and the Assistant solves it. Please help me solve this question. Wrap only the final answer in \boxed{}.

For training vision-language models, we adopt a concise reasoning format inspired by DeepSeek-R1. The model is instructed to place its reasoning inside `<think>` and `</think>` tags and to wrap the final answer in `\boxed{}`. Compared with the Mulberry prompt used in MINT-CoT, this format is simpler and imposes fewer structural constraints on the model output. In the appendix, we further analyze how this prompt design affects learning dynamics and final performance.

DeepSeek-R1's system prompt

You first think through the reasoning process as an internal monologue, enclosed within <think> </think> tags. Then, provide your final answer enclosed within \boxed{}.

This prompt is more concise and straightforward than the Mulberry-style prompt used in the MINT-CoT dataset, which decomposes the response into multiple explicit sections, including image description, rationale, step-by-step reasoning, and final answer. While such a structured format may encourage detailed multimodal reasoning, it also imposes a more complex output schema.

Mulberry's system prompt

Generate an image description based on the question.
Then, provide a rationale to analyze the question.
Next, generate a step-by-step reasoning process to solve the problem. Ensure the steps are logical and concise.
Finally, provide a concise summary of the final answer in the following format: The final answer is: xxx.
Format your response with the following sections, separated by ###:
Image Description:
Rationales:
Let's think step by step.
Step 1:
Step 2:
...
The final answer is:

A.3 Additional experimental results

Figure 5 presents a category-level comparison of the base model, GRPO, and RGPO on MathVista. RGPO achieves the highest overall accuracy, improving from 36.7 for the base model and 39.6 for GRPO to 40.7. The gains are consistent across several reasoning categories, including geometry reasoning, algebraic reasoning, geometry problem solving, and textbook question answering. The improvement is especially pronounced on textbook question answering, where RGPO reaches 53.2, outperforming GRPO by 6.4 points and the base model by 8.0 points.

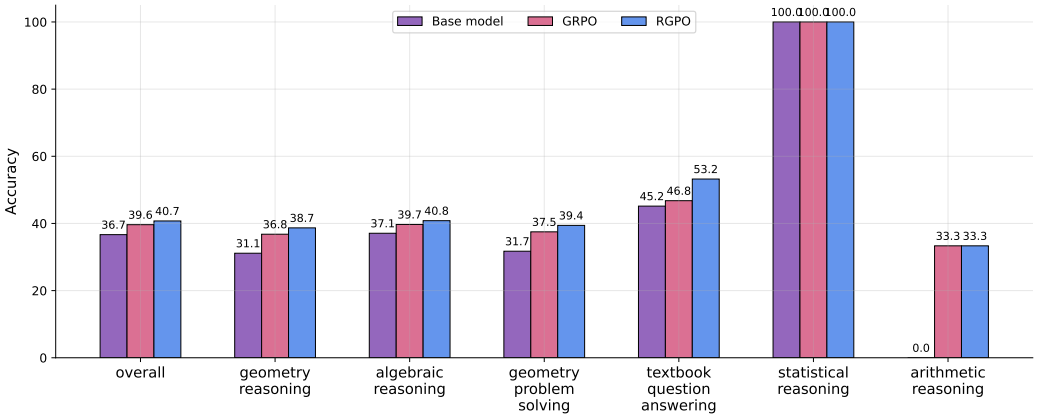


Figure 5: **Category-wise performance comparison on MathVista.** We compare the base model, GRPO, and RGPO across different mathematical reasoning categories. RGPO achieves the best overall accuracy and consistently improves over both the base model and GRPO on most categories, with the largest gain observed on textbook question answering.

This suggests that rationale-guided refinement is particularly helpful for problems requiring structured interpretation and multi-step reasoning. On statistical reasoning, all methods achieve perfect accuracy,

while on arithmetic reasoning both GRPO and RGPO improve over the base model. Overall, these results show that RGPO provides a stronger optimization signal than standard RLVR-style training, leading to better no-hint reasoning performance across most MathVista categories.

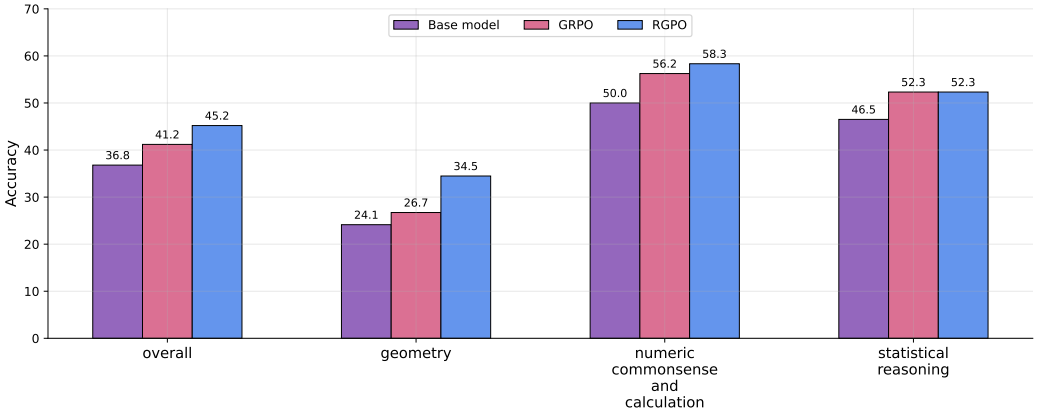


Figure 6: **Category-wise performance comparison on MMStar:** We compare the base model, GRPO, and RGPO across the mathematical reasoning categories in MMStar. RGPO achieves the highest overall accuracy and improves over both baselines on geometry and numeric commonsense/calculation tasks, while matching GRPO on statistical reasoning.

Figure 6 reports the category-wise performance of the base model, GRPO, and RGPO on MMStar. RGPO achieves the best overall accuracy, improving from 36.8 for the base model and 41.2 for GRPO to 45.2. The largest gain appears in geometry, where RGPO reaches 34.5, outperforming GRPO by 7.8 points and the base model by 10.4 points. RGPO also improves numeric commonsense and calculation accuracy to 58.3, compared with 56.2 for GRPO and 50.0 for the base model. On statistical reasoning, RGPO matches GRPO at 52.3, with both methods outperforming the base model. These results show that RGPO provides consistent benefits on MMStar, especially for visually grounded and calculation-heavy reasoning tasks where rationale-guided refinement can help convert failed attempts into more informative training signals.

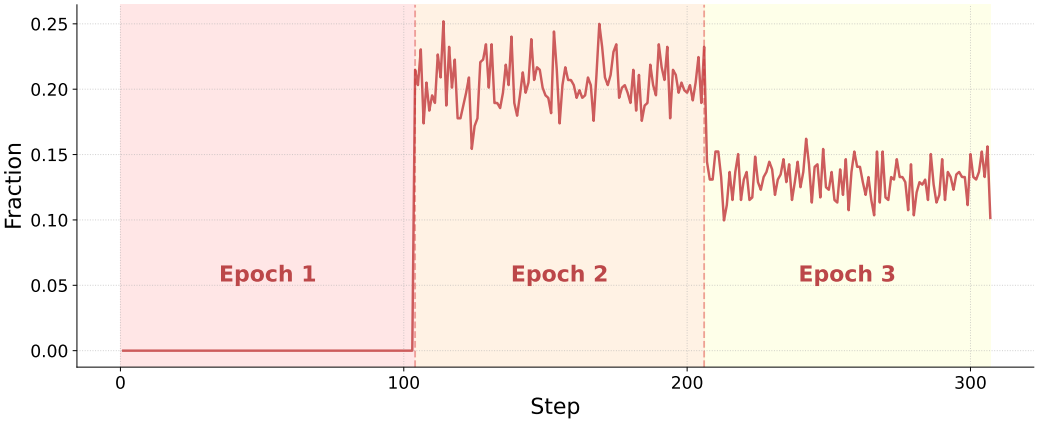


Figure 7: **Fraction of training examples receiving rationale guidance:** The hint rate remains zero during the probing epoch, rises when RGPO begins guided refinement, and decreases as the model learns to solve more examples independently.

Figure 7 illustrates how RGPO adaptively controls the use of rationale guidance during training. In the first epoch, no hints are provided, allowing the algorithm to estimate the model's unguided problem-solving ability. Once rationale-guided refinement is activated in the second epoch, the non-empty hint fraction rises to around 20%, indicating that only a minority of examples require scaffolded assistance. By the third epoch, this fraction drops to roughly 15%, suggesting that many examples that previously required guidance can now be solved without hints. This behavior supports

the central design of RGPO: rationales are not used as always-on supervision, but as temporary scaffolds for difficult examples. As training progresses, the model becomes less dependent on external rationale guidance, making full-solution exposure for every sample unnecessary.

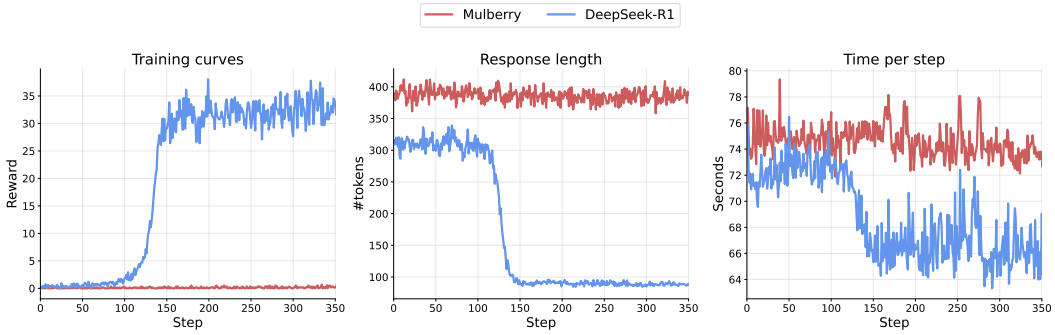


Figure 8: **Effect of prompt format on training dynamics and efficiency.** We compare the Mulberry-style structured prompt with the simpler DeepSeek-R1-style prompt in terms of reward, response length, and runtime per training step. The DeepSeek-R1-style prompt leads to substantially better learning while producing shorter responses and reducing training time.

In Figure 8, we show that prompt format has a significant effect on both optimization and efficiency. The Mulberry-style prompt requires image descriptions, rationales, step-by-step sections, additional delimiters, and line breaks, produces much longer responses throughout training, typically around 380–400 tokens. Despite this higher generation cost, its reward remains nearly flat, suggesting that the complex output schema makes the format difficult to learn and does not translate into better task performance without additional supervision (e.g. from SFT loss in the AHPO baseline in the main paper). In contrast, the DeepSeek-R1-style prompt shows a clear reward increase after training begins to improve, while its response length drops sharply to below 100 tokens, indicating that the model learns a more concise and effective reasoning format. This reduction in response length also lowers the runtime per step, making training more efficient. These results motivate our use of the simpler DeepSeek-R1-style prompt: it reduces formatting burden, avoids unnecessary verbosity, and enables more stable learning without requiring extra supervised fine-tuning to enforce the response structure.

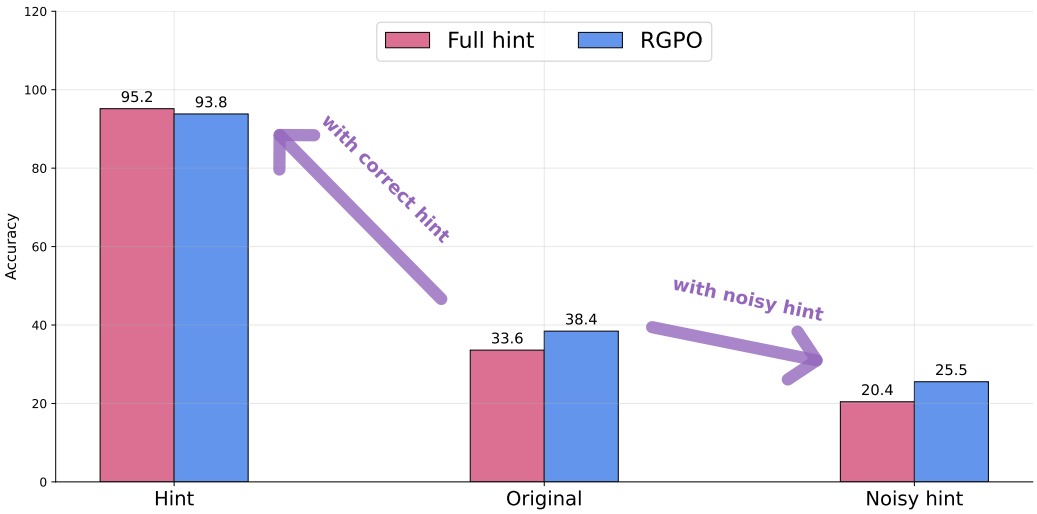
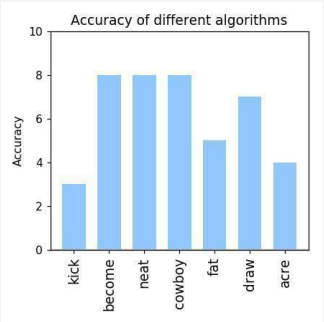


Figure 9: **Robustness to helpful and noisy rationale hints on MMMath:** We compare RGPO with a full-hint baseline under three evaluation settings: correct rationale hints, the original no-hint benchmark, and shuffled noisy hints. While the full-hint baseline performs slightly better when the correct CoT is provided, RGPO achieves higher accuracy in the original and noisy-hint settings, indicating stronger robustness and less dependence on external rationales.

Figure 9 evaluates whether training with rationale guidance leads to genuine reasoning improvement or merely teaches the model to rely on hints. On the left, both methods are evaluated with the correct

CoT appended to the question as a rationale hint. The full-hint baseline achieves the highest accuracy, 95.2, slightly outperforming RGPO at 93.8, which is expected because the baseline is optimized to maximally exploit the provided hint. However, this advantage disappears in the original no-hint setting, where RGPO improves over the full-hint baseline from 33.6 to 38.4. The difference becomes more pronounced under noisy hints, where the CoT is shuffled and appended with an instruction that the model may use it if helpful. In this setting, the full-hint baseline drops to 20.4, suggesting that it tends to trust and follow the misleading rationale rather than solve the problem independently. RGPO achieves a higher score of 25.5, indicating better resistance to corrupted guidance. These results support the motivation of RGPO: rationales should serve as temporary scaffolds during training, not as permanent dependencies.

By selectively using hints only after failure and optimizing for refined solutions, RGPO improves no-hint reasoning while reducing over-reliance on potentially noisy rationales from MMMath [64]. This aligns with recent concerns about over-privileging where they can leak rationales and degrade validation performance, while trained models may overuse unnecessary rationales [65, 66]. Always exposing a full or noisy CoT creates a similar shortcut, encouraging the model to rely on the rationale rather than solve independently. RGPO mitigates this risk by providing hints only after unguided failure, verifier-filtering guided outputs, and distilling accepted responses back to the no-hint prompt.



Correct answer: C

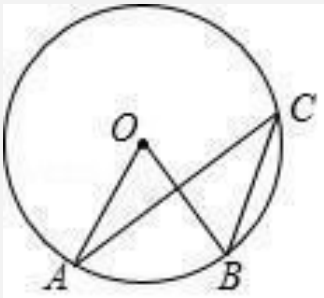
Question: *What is the sum of the accuracies of the algorithms fat and acre?*
Choices:
(A) 8
(B) 15
(C) 9
(D) 12

Base model: D. 12

GRPO: <think> The question asks for the sum of the accuracies of the algorithms fat and acre. To find this, I need to add the values of these two algorithms. </think>

\boxed{D}

RGPO: <think> To find the sum of the accuracies of the algorithms fat and acre, I need to add their values. From the bar chart, the accuracy of fat is 5 and the accuracy of acre is 4. So, the sum is 5 + 4 = 9. Therefore, the correct answer is C.</think>
\boxed{C}



Correct answer: D

Question: *As shown in the figure, points A, B, and C are on ⊙O, if ∠C = 35.0, then ∠AOB = ()*
Choices:
(A) 17.5°
(B) 35°
(C) 60°
(D) 70°

Base model: To determine the measure of ∠AOB, we can use the properties of angles subtended by the same arc in a circle. Specifically, the measure of an angle is equal to half the measure of the arc it subtends.

Given:
- ∠C = 35.0°
- Points A, B, and C are on ⊙O

Since ∠C is subtended by the arc AB, we can find ∠AOB by dividing the measure of ∠C by 2:

∠AOB = 1/2 ∠C = 1/2 * 35.0° = 17.5°

Therefore, the measure of ∠AOB is 17.5°.

The correct answer is:
A. 17.5°

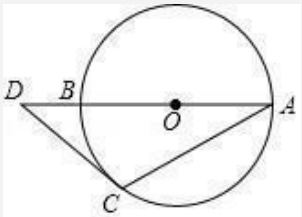
GRPO: <think> To find ∠AOB, we need to use the properties of angles subtended by the same arc. Since ∠C is given as 35°, and ∠AOB is subtended by the same arc, ∠AOB = ∠C = 35°. </think>

\boxed{B}

RGPO: <think> To find ∠AOB, we need to use the properties of angles subtended by the same arc. Since ∠C is given as 35°, and ∠AOB is subtended by the same arc, we can use the formula ∠AOB = 2 * ∠C to find the measure of ∠AOB.</think>
\boxed{D}

Figure 10: **Qualitative comparison of model responses:** Example outputs from the base model, GRPO, and RGPO.

We provide qualitative examples comparing the responses generated by the base model, GRPO, and RGPO in Figure 10 and Figure 11. Compared with the base model, GRPO and RGPO often yield more focused reasoning processes. Moreover, by using rationales as temporary scaffolds, RGPO is able to identify the relevant solution path and answer questions more concisely.



Correct answer: B

Question: *As shown in the figure, AB is the diameter of ⊙O, point D is on the extended line of AB, passing point D is the tangent of ⊙O, and the tangent point is C, if ∠A = 25.0, then ∠D = ()*
Choices:
(A) 25°
(B) 40°
(C) 50°
(D) 65°

Base model: <think> To solve this problem, we need to find the measure of ∠D. We know that AB is the diameter of the circle, so ∠ACB is a right angle (90°). Since CD is a tangent to the circle, ∠D is an inscribed angle that subtends the same arc as ∠ACB. Therefore, ∠D = 1/2 ∠ACB. We can use the given angle ∠A = 25° to find ∠D. </think> <answer> C </answer>

GRPO: <think> To find ∠D, we need to use the properties of tangents and the fact that AB is the diameter of the circle. Since AB is the diameter, ∠ACB is a right angle (90°). Given that ∠A = 25°, we can find ∠ACB by subtracting ∠A from 90°. Then, we can use the fact that the sum of angles in a triangle is 180° to find ∠D. </think>

\boxed{C}

RGPO: <think> To find ∠D, we need to use the properties of tangents and the fact that AB is the diameter of the circle. Since AB is the diameter, ∠ACB is a right angle (90°). Given that ∠A = 25°, we can find ∠ACB by subtracting ∠A from 90°. Then, we can use the fact that the sum of angles in a triangle is 180° to find ∠D. </think>

\boxed{B}

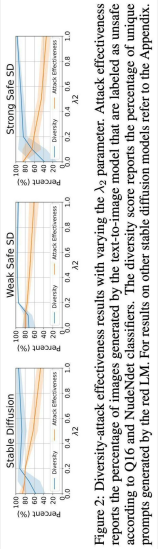


Figure 2: Diversity-attack effectiveness results with varying the λ_2 parameter. Attack effectiveness reports the percentage of images generated by the text-to-image model that are labeled as unsafe according to Q16 and NudeNdet classifiers. The diversity score reports the percentage of unique prompts generated by the red LM. For results on other stable diffusion models refer to the Appendix.

Correct answer: A

Question: *In which of the following value ranges of λ2 does the percentage of Attack Effectiveness begin to be lower than that of Diversity?*
Choices:
(A) 0.0 - 0.2
(B) 0.2 - 0.4
(C) 0.4 - 0.6
(D) 0.6 - 0.8

Base model: B

GRPO: <think>
To determine the value range of λ2 where the percentage of Attack Effectiveness begins to be lower than that of Diversity, we need to analyze the graphs in Figure 2. The graphs show the percentage of images generated by the text-to-image model that are labeled as unsafe according to Q16 and NudeNdet classifiers (Attack Effectiveness) and the percentage of unique prompts generated by the red LM (Diversity) as a function of λ2.

By examining the graphs, we can observe the following:
- At λ2 = 0.0, the percentage of Attack Effectiveness is higher than that of Diversity.
- As λ2 increases, the percentage of Attack Effectiveness decreases, approaching the value of Diversity.
- At λ2 = 0.8, the percentage of Attack Effectiveness is lower than that of Diversity.

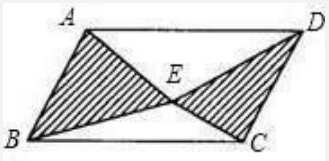
Therefore, the correct answer is D.
</think>
\boxed{D}

RGPO: <think>
To determine the correct value range for λ2 where the percentage of Attack Effectiveness begins to be lower than that of Diversity, we need to analyze the graphs in Figure 2. The graphs show the percentage of images generated by the text-to-image model that are labeled as unsafe according to Q16 and NudeNdet classifiers (Attack Effectiveness) and the percentage of unique prompts generated by the red LM (Diversity) as a function of λ2.
\boxed{A}

Figure 11: **Qualitative comparison of model responses:** Example outputs from the base model, GRPO, and RGPO.

A.4 Hardware configuration

All experiments were conducted on high-performance machines equipped with Intel Xeon CPUs and NVIDIA GPUs, selected to accommodate varying computational needs and optimize job priority scheduling across different tasks. Specifically, we utilized two machine configurations: (1) INTEL(R) XEON(R) PLATINUM 8592+ with 2.0Ti RAM and H200 SXM 141GB GPUs and (2) Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz with 503Gi RAM and NVIDIA A100 SXM4 80GB GPUs. Although different GPU types were used to balance workload priorities, we ensured that all comparisons across methods were performed on the same hardware configuration for a given model and dataset to eliminate hardware-induced variability and maintain consistency and fairness in evaluation.



Correct answer: B

Question: *As shown in the figure, E is any point in ▱ABCD, if S~quadrilateral ABCD~ = 6.0, then the area of the shaded part in the figure is ()*
Choices:
(A) 2
(B) 3
(C) 4
(D) 5

Base model: To determine the area of the shaded part in the figure, we can use the properties of parallelograms and triangles.

1. **Identify the parallelogram**: The figure shows a parallelogram with vertices A, B, C, and D. The shaded part is a triangle within this parallelogram.

2. **Calculate the area of the parallelogram**: The area of a parallelogram is given by the formula \(A = base \times height\). Since the parallelogram is a rectangle, its base and height are equal. Therefore, the area of the parallelogram is \(6 \times 6 = 36\) square units.

3. **Calculate the area of the shaded triangle**: The shaded triangle is a right triangle because it is formed by the intersection of the parallelogram and the rectangle. The area of a triangle is given by the formula \(A = \frac{1}{2} \times base \times height\). Since the parallelogram is a rectangle, its height is equal to the length of the parallelogram's base. Therefore, the area of the shaded triangle is \(\frac{1}{2} \times 6 \times 6 = 18\) square units.

4. **Calculate the area of the shaded part**: The area of the shaded part is the area of the parallelogram minus the area of the triangle. Therefore, the area of the shaded part is \(36 - 18 = 18\) square units.

5. **Select the correct option**: The area of the shaded part is 18 square units, which corresponds to option C.

GRPO: <think> To find the area of the shaded part, we need to consider the properties of the parallelogram and the triangles formed by the intersection of the diagonals. Since E is any point in ▱ABCD, the shaded part is a part of the parallelogram. The area of the shaded part can be calculated by finding the area of the parallelogram and subtracting the areas of the two triangles formed by the intersection of the diagonals.</think>

\boxed{C}

RGPO: <think> To find the area of the shaded part, we need to understand the relationship between the shaded and non-shaded parts. Since E is an intersection point of the diagonals of ▱ABCD, the shaded part is a trapezoid with bases AB and CD and height DE. The area of a trapezoid is given by the formula: Area = (1/2) * (sum of the bases) * height. Therefore, the area of the shaded part is (1/2) * (sum of bases AB and CD) * height DE. </think>

\boxed{B}

Figure 12: **Qualitative comparison of model responses:** Example outputs from the base model, GRPO, and RGPO.

B Limitations

RGPO relies on the availability of reference solutions; its effectiveness therefore depends on both the quality of those solutions and the granularity of the segmentation. If the revealed prefix contains too much answer-specific information, the method may encourage shortcut learning or memorization rather than transferable reasoning; if it contains too little or is stylistically far from the model's own reasoning distribution, it may fail to rescue hard examples from sparse-reward collapse. Moreover, the reward-gated acceptance rule only guarantees improvement under the available verifier, not genuine reasoning quality, so RGPO can still inherit verifier bias, reward hacking, or spurious formatting gains. The adaptive schedule also introduces additional compute from guided rollouts and auxiliary SFT updates, making the method more expensive than a single-pass PPO or GRPO baseline. However, we keep the number of rollouts per prompt the same as in the GRPO baseline to ensure a fair comparison.

C Broader Impacts

RGPO aims to make reasoning-oriented post-training more sample-efficient by using reference rationales as adaptive training-time scaffolds, which may help smaller language and vision-language models learn from sparse verifiable rewards without relying as heavily on large teacher models, rejection sampling, or fully formatted supervised traces; this could benefit education, scientific analysis, software assistance, mathematical problem solving, and multimodal question answering. However, stronger reasoning ability is inherently dual-use: the same improvements that help benign problem solving may also assist harmful planning, automated persuasion, misinformation generation, cyber misuse, or other strategic tasks. RGPO also improves verifier reward rather than guaranteeing

faithful reasoning, so generated rationales may remain incomplete, misleading, or optimized for the reward function rather than for truth, especially when verifiers or reference solutions contain biases, shortcuts, or annotation artifacts. These risks suggest that RGPO should be paired with careful verifier design, dataset auditing, robustness and out-of-distribution evaluation, and appropriate deployment safeguards for high-capability models. The method also adds compute overhead through guided rollouts and auxiliary distillation, although this cost may be offset in some settings if adaptive hinting improves sample efficiency or enables smaller models to reach competitive reasoning performance.