ARPO: Answer-Refined Policy Optimization for Learning from Hard Instances in Group-Relative RLVR

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) and group-relative objectives such as Group Relative Policy Optimization (GRPO) have delivered strong gains for improving the reasoning ability of large language models on complex mathematical benchmarks. However, progress is often bottlenecked by hard instances that expose missing reasoning skills. Under strict binary verification, these instances frequently yield no verified successes within a rollout group, causing group-relative normalization to provide little usable gradient signal and leading to a learning cliff. We propose Answer-Refined Policy Optimization (ARPO), a GRPO-style algorithm for recovering informative updates on such all-zero groups. When a rollout group contains no verified success, ARPO uses the training-time reference answer to prompt the current policy to produce a standardized step-by-step reasoning trace that reaches the correct final answer and passes verification. The recovered trace is then inserted back into the group as a verified positive rollout, so that the same verifier-based RLVR update can proceed with a non-degenerate learning signal under the original problem prompt. Experiments on diverse mathematical reasoning benchmarks, from exam-style datasets to olympiad-level problems, show that ARPO consistently reduces training stagnation in sparse-success regimes and improves final solution rates under strict verifier feedback.

1 Introduction

Large language models (LLMs) have recently achieved strong performance on complex reasoning tasks such as competition-style mathematics, olympiad problems, and formal exam benchmarks. When answers are programmatically checkable, *reinforcement learning with verifiable rewards*

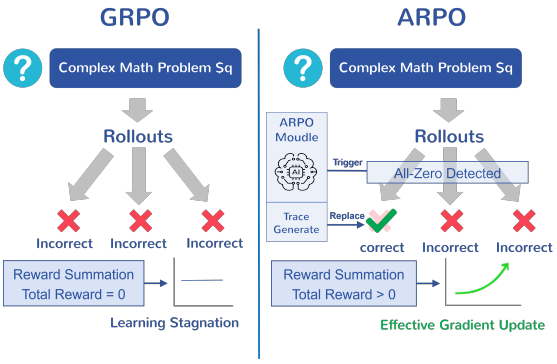

Figure 1: GRPO vs. ARPO on hard instances. When all sampled rollouts in a group are incorrect, GRPO receives zero group reward and provides no useful learning signal. ARPO detects the all-zero condition, generates a verified correct rollout using the reference answer as guidance, and injects it into the rollout group, thereby recovering a non-zero reward signal and enabling effective gradient updates.

(RLVR) has emerged as a central paradigm, directly optimizing against automated verifiers instead of human preference models (Wen et al., 2025; Wu and Choi, 2025). RLVR-style training underpins many recent reasoning-focused systems and yields substantial gains on math and other structured problem-solving tasks.

Despite this success, RLVR still struggles precisely on the most informative portion of the data, namely hard instances that expose missing reasoning skills. In typical RLVR pipelines, the verifier only credits exactly correct final answers, often yielding binary rewards $\{0, 1\}$. Early in training, or on especially challenging questions, the policy rarely reaches a correct solution, meaning these hard instances contribute no explicit positive signal about what a correct reasoning trajectory looks like. This is not merely an issue of sparse rewards. It is a supervision gap on the very failures we most need to learn from.

This gap becomes particularly acute in group-relative objectives such as GRPO. When every sam-

pled rollout for a hard instance fails verification, the group provides no successful trajectory to anchor learning. In the c ommon binary-reward setting, the group statistics collapse and the normalized advantages become uninformative, yielding an approximately zero policy gradient. Consequently, hard instances can become effectively invisible to optimization, producing a learning cliff where performance saturates because training stops extracting signal from a substantial fraction of difficult data (Zhang et al., 2025b; Wu and Choi, 2025).

Recent RLVR advances improve stability and sample efficiency by modulating difficulty or adding guidance. Difficulty-aware curricula and scaffolded frameworks adjust training distributions toward solvable yet challenging problems (Zhang et al., 2025b; Zhang and Zuo, 2025). Off-policy teacher guidance, such as prefix- or hint-based approaches including Prefix-RFT, StepHint, or BREAD, seeds successful completions but introduces a mismatch between teacher-provided contexts and student-generated trajectories (Yan et al., 2025; Huang et al., 2025; Zhang et al., 2025a,c). Meanwhile, variance-aware or advantage-shaping methods, such as AGPO (Li et al., 2025), NGRPO (Nan et al., 2025) or RL-ZVP (Le et al., 2025), can adjust gradients in degenerate groups, but they still do not produce a verifiable positive rollout when the policy fails on every attempt. They reshape credit assignment without supplying an explicit correct solution trajectory for the hard instance.

In this work, we propose Answer-Refined Policy Optimization (ARPO), a simple augmentation for the specific failure mode where a rollout group contains no verified success. During training, we detect such all-zero groups and retrieve the training-time reference answer already available in the dataset. We then prompt the current policy to construct a step-by-step reasoning trajectory that reaches that answer. The resulting answer-refined traces are filtered by the same external verifier, standardized, and, when successful, used to replace one failed rollout in the group. In this way, ARPO converts otherwise uninformative all-zero groups into groups that contain at least one verifier-passing trajectory, allowing the standard group-relative update to recover a useful learning signal on exactly the hard instances that would otherwise be missed.

We evaluate ARPO on a diverse set of mathematical reasoning benchmarks and across multiple backbone models. Empirically, ARPO prevents training from stalling on difficult problems where strictly verified correct solutions are rare, and it consistently improves final solve rates by turning hard failures into verifiable successes via explicit positive rewards. Our main contributions are summarized as follows:

- We characterize signal starvation in group-relative RLVR on hard instances, and introduce the *all-zero rate* to detect groups with no verified successes.
- We introduce ARPO, a GRPO extension that activates only on all-zero groups and uses training-time reference answers to recover verifier-passing trajectories from the current policy, turning hard failures with no positive signal into informative group-relative updates.
- We validate ARPO on challenging math benchmarks across multiple model sizes, showing consistent gains under strict binary verification, especially on settings with sparse verified successes and high all-zero rates.

2 Related Work

Reinforcement learning from verifier reward. The success of DeepSeek-R1 and related systems has established reinforcement learning with verifiable rewards (RLVR) as a central paradigm for enhancing the reasoning capabilities of large language models (Wen et al., 2025; Wu and Choi, 2025). In RLVR, models are trained using feedback from an external verifier that typically provides a sparse binary reward based on final answer correctness. Subsequent work improves stability and sample efficiency by debiasing and analyzing R1 style training dynamics (Liu et al., 2025; Yu et al., 2025), or by designing more informative rewards, such as length penalties to mitigate overthinking (Aggarwal and Welleck, 2025; Jin et al., 2025; Chen et al., 2024) and token level signals to provide denser feedback (Wang et al., 2025b,a; Ouyang et al., 2025). Difficulty aware curricula and scaffolded variants of GRPO further adapt the training distribution toward problems that are challenging yet still solvable, introducing structured guidance to elicit better reasoning from the model (Zhang et al., 2025b; Zhang and Zuo, 2025).

Nevertheless, these approaches still leave a supervision gap on hard instances that expose missing reasoning skills. Under strict verification, a rollout group may contain no verified success, so training lacks a positive trajectory that demonstrates how to solve the instance. In group based objectives, the

absence of successes can also collapse normalized learning signals and yield little effective gradient, causing these hard instances to contribute minimally to optimization (Wu and Choi, 2025). Our approach targets this failure mode directly by detecting such instances and converting them into verifiable successes through answer refined trajectories, while keeping the underlying RLVR objective and optimization pipeline unchanged.

RLVR with guided trajectories. Another line of work mitigates the learning cliff by introducing guided trajectories from stronger teachers or expert solutions. A common strategy is to expose the model to partial or complete correct reasoning and train it to continue or refine these trajectories (Yan et al., 2025; Huang et al., 2025; Zhang et al., 2025a,c; Ma et al., 2025). Representative examples include LUFFY, which mixes expert rollouts with policy-generated samples in the same batch (Yan et al., 2025); Prefix-RFT, which employs a decaying prefix schedule to gradually reduce reliance on teacher steps (Huang et al., 2025); StepHint, which provides stepwise hints at multiple granularities (Zhang et al., 2025a); and BREAD, which expands exploration from expert anchors through branched rollouts (Zhang et al., 2025c).

These methods are effective at increasing the frequency of verified successes on difficult instances, but they do so by injecting intermediate solution structure or expert guidance into the rollout process. ARPO instead studies a more minimal form of guidance: it uses only the final answer already available in the training data and asks the current policy itself to recover a complete reasoning trajectory that reaches that answer and passes verification. This formulation lets us isolate whether final-answer-based trajectory recovery alone is sufficient to repair all-zero groups under the same verifier-based RLVR backbone, without introducing expert intermediate steps or auxiliary teacher trajectories. Accordingly, our main comparisons focus on methods evaluated within a shared GRPO-style setup, where the training data, verifier, decoding configuration, and optimization schedule are matched.

3 Method

In this section, we formalize RLVR within the GRPO framework, identifying "hard instances" as rollout groups that lack verified successes. These all-zero groups yield vanishing gradients under group-relative normalization. We introduce ARPO to address this failure mode by recovering verifier-passing trajectories for all-zero groups and feeding them back into group-relative RLVR updates, thereby restoring informative learning signals on otherwise stagnant instances.

3.1 Preliminaries

We consider RLVR for reasoning problems such as mathematics and coding. Let $\mathcal{D} = \{(q, a^\star)\}$ denote a dataset of problems with verifiable ground truth answers, where q is an input question and $a^\star$ is the reference answer. A language model policy π_θ maps q to an output trajectory o that contains intermediate reasoning and a final answer. An external verifier computes a scalar reward $R(o; q, a^\star) \in \{0, 1\}$, which is 1 if the extracted final answer is accepted and 0 otherwise.

Following recent work, we adopt a group-based policy optimization objective such as GRPO. For each problem q, the policy samples a group of N rollouts $G(q) = \{o_i\}_{i=1}^N$, $o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, each with reward $R_i = R(o_i; q, a^\star)$. GRPO uses the group mean $\mu_G = \frac{1}{N}\sum_{i=1}^N R_i$ and variance $\sigma_G^2 = \frac{1}{N}\sum_{i=1}^N (R_i - \mu_G)^2$ to define a normalized per trajectory advantage

$$\hat{A}_i = \frac{R_i - \mu_G}{\sigma_G + \epsilon_{\text{std}}}, \tag{1}$$

where ϵ_{std} is a small constant. These advantages are then used in a standard clipped PPO style token level objective, with token level importance ratios $r_{i,t}(\theta)$ computed between π_θ and $\pi_{\theta_{\text{old}}}$.

We say that a problem q is hard for the current policy $\pi_{\theta_{\text{old}}}$ if the probability of obtaining a verified correct solution is extremely low. Let $p_{\text{succ}}(q) = \mathbb{E}[R(o; q, a^\star) \mid o \sim \pi_{\theta_{\text{old}}}]$ denote the success probability for q. In practice, hard instances often manifest as rollout groups that contain no verified success.

We call $G(q)$ a *no success group* if no rollout in the group passes verification, namely $\sum_{i=1}^N R_i = 0$. In the binary reward setting used throughout this work, a no success group is equivalent to every rollout receiving zero reward, that is $R_i = 0$ for all i. In this regime, the group statistics collapse to $\mu_G = 0$ and $\sigma_G = 0$, so every normalized advantage becomes exactly zero after adding the stabilizer, $\hat{A}_i = 0$ for all i. Consequently, the contribution of this problem to the gradient of the GRPO objective is essentially zero, and these hard

Algorithm 1 RLVR with ARPO

Require: Dataset $\mathcal{D}$, Policy π_θ, Verifier R, Group size N, ARPO samples M

Ensure: Optimized policy parameter θ

while training **do**
  Sample a batch of problems $\mathcal{B} \sim \mathcal{D}$
  for each $(q, a^\star) \in \mathcal{B}$ **do**
    Sample group $G = \{o_1, \ldots, o_N\}$ where $o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$
    Compute rewards $R_i \leftarrow R(o_i; q, a^\star)$ for $i = 1, \ldots, N$
    if $\sum_{i=1}^{N} R_i = 0$ **then**
      Construct answer-refined prompt $x_{\text{ans}} \leftarrow T_{\text{ans}}(q, a^\star)$
      Sample candidates $\{o_j^{\text{ans}}\}_{j=1}^{M} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x_{\text{ans}})$
      $\mathcal{S} \leftarrow \{o_j^{\text{ans}} \mid R(o_j^{\text{ans}}; q, a^\star) = 1\}$
      if $\mathcal{S} \neq \emptyset$ **then**
        Select $\tilde{o}$ uniformly from $\mathcal{S}$
        Standardize format of $\tilde{o}$
        $o_N \leftarrow \tilde{o}$; $R_N \leftarrow 1$
      end if
    end if
    Compute group mean μ_G and variance σ_G of $\{R_i\}_{i=1}^{N}$
    Compute advantages $\hat{A}_i \leftarrow \frac{R_i - \mu_G}{\sigma_G + \epsilon_{\text{std}}}$
  end for
  Update θ using GRPO objective with advantages $\{\hat{A}\}$
end while

instances provide no explicit positive trajectory to learn from in the current update.

3.2 Answer-Refined Policy Optimization

We now introduce ARPO, a simple wrapper around RLVR that turns all-zero groups into informative updates. Conceptually, ARPO consists of three stages applied only to problems that induce all-zero groups (as illustrated in Figure 2): (i) detecting all-zero groups, (ii) generating answer-refined trajectories from the current policy, and (iii) standardizing and replacing one trajectory in the group with a successful one, treating it as a rollout with positive reward. The complete training procedure integrating ARPO with GRPO is summarized in Algorithm 1.

3.2.1 Detecting All-Zero Groups

For each training problem $(q, a^\star) \in \mathcal{D}$, the base RLVR algorithm samples a group $G(q) = \{o_1, \ldots, o_N\}$ and computes rewards $R_i = R(o_i; q, a^\star)$ using the external verifier. If at least one rollout in $G(q)$ attains positive reward, ARPO is inactive and we directly apply the standard group-based update on this group. We call $G(q)$ an *all-zero group* if all $R_i = 0$; ARPO is invoked only in this case, where the group-normalized advantages would otherwise collapse to (approximately) zero and the problem would contribute essentially no gradient.

3.2.2 Answer-Conditioned Trajectory Generation

When an all-zero group is detected for $(q, a^\star)$, ARPO first retrieves the corresponding reference answer $a^\star$ from the dataset.

Given $(q, a^\star)$, our goal is to *search for a trajectory that explains how to reach* $a^\star$ starting from q. Let $g(o)$ denote a deterministic extractor that maps a trajectory o to its decoded final answer (e.g., the expression inside the last `\boxed{}`). We write

$$\mathcal{O}_{\text{succ}}(q, a^\star) \;=\; \big\{\, o \;\big|\; g(o) = a^\star \big\} \tag{2}$$

for the set of trajectories whose final answer matches the reference answer. Ideally, we would like to sample from the policy conditioned on being correct,

$$\tilde{\pi}_\theta(o \mid q, a^\star, r = 1) \;\propto\; \tilde{\pi}_\theta(o \mid q)\, \mathbb{I}\{o \in \mathcal{O}_{\text{succ}}(q, a^\star)\}. \tag{3}$$

which reuses the original policy $\pi_\theta(\cdot \mid q)$ but restricts it to trajectories that lead to the reference answer $a^\star$.

Directly sampling from $\tilde{\pi}_\theta(o \mid q, r = 1)$ by naive rejection on $\tilde{\pi}_\theta(o \mid q)$ is infeasible for zero-variance groups, because successful trajectories are extremely rare. Instead, we introduce a tractable proposal distribution $\tilde{\pi}_\theta(o \mid q, a^\star)$ implemented via an answer-refined prompt. Let $T_{\text{ans}}(q, a^\star)$ denote a deterministic transformation that maps $(q, a^\star)$ to a text input containing the original question, the reference answer, and an instruction to produce a detailed step-by-step solution whose final answer equals $a^\star$. We then define

$$\tilde{\pi}_\theta(o \mid q, a^\star) = \tilde{\pi}_\theta\big(o \mid T_{\text{ans}}(q, a^\star)\big), \tag{4}$$

and ARPO draws M answer-conditioned trajectories from the current policy,

$$o_1^{\text{ans}}, \ldots, o_M^{\text{ans}} \;\sim\; \tilde{\pi}_{\theta_{\text{old}}}(o \mid q, a^\star). \tag{5}$$

Each o_m^{ans} is therefore a candidate reasoning path sampled from a prompt-level approximation of

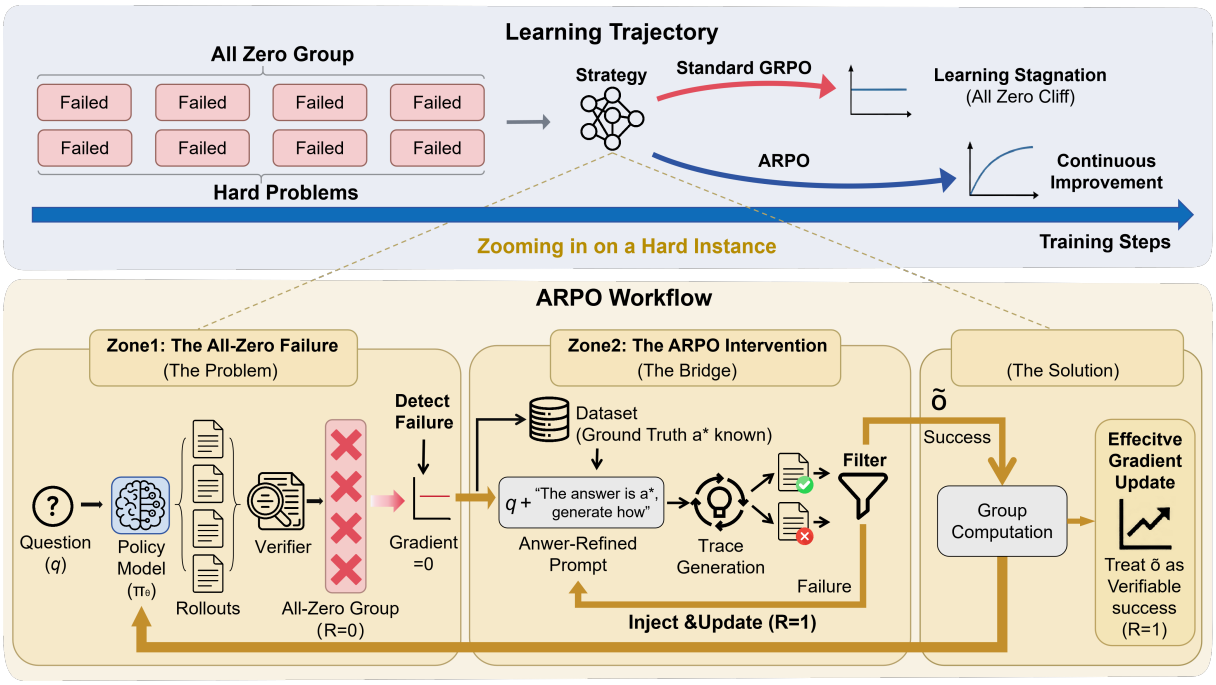

Figure 2: Overview of the ARPO framework. ARPO is activated only on all-zero groups, i.e., rollout groups in which all sampled trajectories fail verification. The figure illustrates the three-stage workflow: (1) detecting an all-zero group, (2) generating answer-refined candidate trajectories using the training-time reference answer, and (3) replacing one failed rollout with a verifier-passing trajectory so that the group-relative update can proceed with a non-zero learning signal.

the ideal success-conditioned distribution $\tilde{\pi}_\theta(o \mid q, a^\star, r = 1)$. Conditioning on the reference answer in this way turns the original "solve from scratch" task into a targeted explanation problem that is substantially easier for a partially trained model, while still allowing the policy to explore diverse trajectories that are consistent with the given final answer.

3.2.3 Standardizing and Replacing ARPO

The answer-refined generations are post-processed before they are incorporated into the RLVR update. First, we extract a candidate final answer $\hat{a}_m$ from each o_m^{ans} using a simple parser and re-run the external verifier on $(q, \hat{a}_m)$. This yields verifier rewards $R(o_m^{\text{ans}}; q, a^\star) \in \{0, 1\}$. We define the set of answer-refined trajectories that pass verification as

$$\mathcal{S}(q, a^\star) = \left\{ o_m^{\text{ans}} \;\middle|\; R(o_m^{\text{ans}}; q, a^\star) = 1 \right\}. \quad (6)$$

If $\mathcal{S}(q, a^\star)$ is empty, ARPO leaves $G(q)$ unchanged in this iteration, and the instance remains an all-zero group. Otherwise, we select one trajectory $\tilde{o} \in \mathcal{S}(q, a^\star)$ (e.g., uniformly at random, with a small implementation-level preference for trajectories that explicitly mark the final answer), and optionally standardize its textual format. ARPO then replaces one failed rollout in the original group with $\tilde{o}$, so that the subsequent group-relative update is computed on a group that now contains a verifier-passing trajectory. The answer-refined prompt is used only to recover $\tilde{o}$; during optimization, the token log-probabilities and PPO ratios of the injected trajectory are still evaluated under the original question prompt q. In this way, ARPO turns an otherwise uninformative all-zero group into a valid training signal while keeping the same verifier-based RLVR update structure.

Likelihood-Based Policy Shaping. Because answer-refined trajectories are proposed under an answer-refined generation context but optimized under the original problem prompt, their likelihood under the two conditions can differ. To account for this prompt-induced mismatch, we compute the log-likelihood gap

$$\Delta(o) = \log \tilde{\pi}_\theta(o \mid q, a^\star) - \log \pi_\theta(o \mid q),$$

and define a clipped shaping weight

$$w(o) = \operatorname{clip}\big(\exp(-\Delta(o)),\, e^{-c},\, e^{c}\big).$$

During optimization, we apply $w(o)$ only to the advantages of injected answer-refined rollouts, while leaving the remaining rollouts unchanged. This

provides a simple likelihood-based calibration that keeps recovered trajectories well aligned with the original RLVR update and prevents any single answer-refined replacement from exerting disproportionate influence.

4 Experiment

We evaluate ARPO on diverse math reasoning benchmarks spanning high-school contests to olympiad-level problems. The experiments show that ARPO improves final solution rates under sparse verifier rewards and that these gains are closely tied to reductions in all-zero groups. We further run ablations to quantify the impact of answer refinement, trajectory recovery, and group-level triggering, and we analyze the additional compute introduced by the conditional re-rollout stage.

4.1 Experimental Setup

We compare ARPO against GRPO and several recent group-relative RLVR variants under a unified training and evaluation protocol. Across settings, ARPO improves final solution rates by recovering learning signal on all-zero groups, while introducing only conditional extra computation when such groups occur. Ablations isolate the contributions of answer refinement, answer-refined trajectory construction, and group-level triggering, clarifying when ARPO most effectively mitigates reward sparsity.

Benchmarks. We evaluate on six math reasoning benchmarks: AIME24, AIME25, AMC23, MATH500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024), covering competition-style problems from high-school to olympiad level. Following common practice, we report average pass@1 over multiple independent runs in a text-only setting, using the same temperature and a maximum generation length that avoids truncating valid solutions. To ensure stable results, we perform 32 runs for AIME24, AIME25, and AMC-23, and 4 runs for other benchmarks. For dataset-level ablations on the replacement ratio, we additionally consider GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021).

Compared Methods. We compare ARPO with RLVR-based baselines implemented in a shared framework. *BaseModel* is the backbone without RLVR fine-tuning, and *GRPO* is standard Group Relative Policy Optimization. We further include Dr.GRPO, GPG (Chu et al., 2025), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and GRPO-lead (Zhang and Zuo, 2025). All methods use the same data, verifier, and decoding settings.

Implementation Details. We fine-tune Qwen2.5-Math-7B (Yang et al., 2024) on the MATH dataset (Hendrycks et al., 2021) with our ARPO algorithm under a standard reasoning-oriented RLVR setup (Liu et al., 2025). To test model-agnosticity, we also train Qwen2.5-Math-1.5B and Qwen2.5-3B with the same training split, preprocessing, and prompt templates as prior RLVR/GRPO-style work (Liu et al., 2025). All experiments run on 8 NVIDIA A100 GPUs using a modified `verl-0.4.0` GRPO implementation. For each backbone and method, we share the same training schedule. ARPO adds no learnable parameters and only introduces lightweight ARPO on all-zero groups, so its computational cost is comparable to GRPO under the same sampling budget. Full hyperparameters, prompt formats, and verifier details are provided in Appendix B.

4.2 Main Results

As shown in Table 1, on Qwen2.5-Math-7B, vanilla GRPO already yields a substantial jump in average pass@1, while recent GRPO-style variants (e.g., Dr.GRPO/GPG/DAPO/GSPO) provide only marginal changes on top of GRPO. ARPO delivers a further improvement, with the largest gains appearing on the hardest benchmarks such as AIME25 and MATH500. This pattern is consistent with ARPO's intended role: it is activated precisely when standard group-relative RLVR produces all-zero groups and recovers a verifier-passing trajectory for those otherwise stagnant instances. Under the same training data, verifier, and decoding setup, these gains indicate that handling all-zero groups more effectively can translate into stronger downstream performance.

We observe the same trend on smaller backbones in Table 2. On Qwen2.5-Math-1.5B, ARPO adds a consistent improvement over GRPO, and on Qwen2.5-3B it improves all evaluated benchmarks with a modest but stable average gain. Overall, the results suggest that ARPO scales smoothly with model size and is most helpful in sparse-success regimes where hard instances otherwise contribute little or no useful gradient.

Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
BaseModel	12.19	4.79	35.23	48.60	15.07	16.33	22.04
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
Dr.GRPO	21.04	8.23	58.59	72.05	28.58	35.89	37.40 (-0.21)
GPG	21.98	9.06	59.61	72.05	27.21	37.67	37.93 (+0.32)
DAPO	21.25	8.75	58.20	72.70	29.50	37.22	37.94 (+0.33)
GSPO	19.38	8.33	60.16	73.00	28.12	37.26	37.71 (+0.10)
GRPO-lead	21.56	9.48	59.06	73.25	29.14	37.07	38.26 (+0.65)
ARPO (ours)	**22.08**	**12.29**	**60.33**	**77.20**	**31.53**	**38.87**	**40.38** (+2.77)

Table 1: Overall performance (pass@1) of different GRPO-style objectives on math reasoning benchmarks using Qwen2.5-Math-7B as the base model.

Model / Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
Qwen2.5-Math-1.5B							
BaseModel	6.87	3.65	30.94	34.95	8.55	21.93	17.82
GRPO	11.35	3.96	46.48	64.85	20.13	29.59	29.39
ARPO (ours)	11.67	5.83	46.72	66.85	22.52	30.19	30.63(+1.24)
Qwen2.5-3B							
BaseModel	2.81	0.73	22.66	48.65	13.69	19.37	17.99
GRPO	5.31	1.56	33.28	63.35	22.89	26.41	25.47
ARPO (ours)	5.89	1.87	35.86	64.05	23.71	28.37	26.63 (+1.16)

Table 2: Performance (pass@1) of ARPO on different Qwen2.5 backbones.

4.3 ARPO Reduces All-zero Reward Groups

To quantify ARPO's effect on the learning cliff, we monitor the *all-zero rate*: the fraction of training problems for which all sampled rollouts receive zero reward, so the corresponding groups contribute no gradient. Figure 3 shows that vanilla GRPO often suffers from signal collapse on challenging benchmarks, especially under limited rollout budgets. For example, with $N = 4$, Qwen2.5-Math-1.5B exhibits an all-zero rate above 22% on MATH. Because ARPO is triggered exactly on such groups, the pre-ARPO all-zero rate also reflects its activation frequency. This frequency varies across tasks: under the same $N = 4$ setting, SVAMP and GSM8K have much lower pre-ARPO all-zero rates (about 4.9% and 9.9%), while MATH contains substantially more such failures. ARPO sharply reduces the all-zero rate across all evaluated settings, consistently pushing it to single digits. ARPO therefore targets cases where standard RLVR provides no learning signal, converting otherwise degenerate groups into valid updates. This reduction in signal starvation matches the downstream gains in our main results.

We further vary the number of rollouts ($N \in \{4, 8, 16\}$) to study the interaction between sample efficiency and signal availability. Increasing the rollout count in vanilla GRPO lowers the all-zero rate but does not remove it; even with $N = 16$, the 1.5B model still shows a 13% rate on MATH. ARPO has the largest relative effect in low-rollout regimes: on MATH with the 1.5B model, it reduces the all-zero rate by roughly 18 percentage points at $N = 4$, versus about 11 points at $N = 16$. This suggests that while more rollouts partially mitigate reward sparsity, ARPO provides a more sample-efficient way to maintain useful gradient signals.

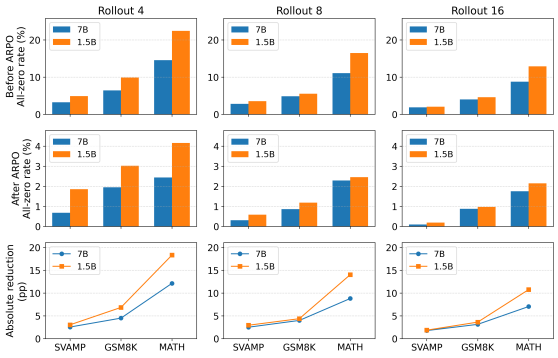

Figure 3: Effect of ARPO on all-zero rate across rollout budgets ($N = 4, 8, 16$). The top two rows show all-zero rates before and after ARPO, while the bottom row displays the absolute reduction.

4.4 Quality of Answer-Refined Trajectories

Validation via Supervised Fine-Tuning. To assess the quality of the reasoning trajectories synthesized by our answer-refined mechanism, we conduct a SFT experiment. Using **Qwen2.5-Math-7B** as the base model, we generate reasoning chains

Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔBase
BaseModel	12.19	4.79	35.23	48.60	15.07	16.33	22.04
SFT	14.17	7.08	42.39	69.85	24.72	23.51	30.29(+8.25)

Table 3: Performance comparison between BaseModel and SFT across reasoning benchmarks.

Methods / ARPO ratio	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
BaseModel	12.19	4.79	35.23	48.60	15.07	16.33	22.04
0% (GRPO)	20.94	8.44	58.98	72.20	27.76	37.33	37.61
25%	21.04	8.96	59.11	74.85	28.12	37.41	38.25 (+0.64)
50%	21.35	10.62	59.47	76.30	30.88	38.02	39.44 (+1.83)
100% (Ours)	22.08	12.29	60.33	77.20	31.53	38.87	40.38 (+2.77)

Table 4: Ablation on the fraction of all-zero instances replaced by answer-refined rollouts during GRPO training on Qwen2.5-Math-7B: 0% is vanilla GRPO, and 100% is full ARPO.

for the MATH dataset by prompting the model with ground-truth answers, mimicking the exploration phase of ARPO, and then fine-tune the base model on these trajectories with cross-entropy loss.

As summarized in Table 3, this SFT procedure raises the average accuracy from 22.04% to 30.29%, with especially large gains on MATH500. This confirms that providing the final answer effectively elicits correct reasoning that was previously latent in the model, turning difficult problems into useful training data. Nevertheless, SFT still lags behind the full ARPO reinforcement learning result (40.38% average), suggesting that the active exploration and optimization in RLVR remain crucial beyond simple behavior cloning.

Quality Verification with LLM Judge. A central concern with answer-refined generation is whether the model truly reasons through the steps or just generates a plausible path to the provided solution. We examine this through large-scale validation on the full MATH dataset using Qwen2.5-Math-7B.

Using the same ARPO exploration prompt as in training, we provide the ground-truth answer and generate reasoning trajectories for all training instances, then use **GPT-5.1** as an automated judge to assess their logical validity. Among the 12,485 samples that follow the required format and reach the provided answer, the judge finds that **90.67%** (11,319/12,485) are logically sound. This suggests that ARPO predominantly elicits valid latent reasoning rather than superficial hindsight explanations. Details of the prompt and evaluation protocol are provided in Appendix A.2.1.

4.5 Ablation Study

To isolate the effect of ARPO, we ablate the fraction of *all-zero* instances that trigger answer refinement on Qwen2.5-Math-7B. We define a replacement ratio $r \in \{0\%, 25\%, 50\%, 100\%\}$ over all-zero instances, i.e., rollout groups whose sampled trajectories all receive zero reward. For each r, we apply answer refinement to $r\%$ of such instances and replace one failed trajectory in each selected group with a verifiable answer-refined rollout, leaving the rest unchanged. When $r = 0\%$, the method reduces to vanilla GRPO; when $r = 100\%$, it becomes full ARPO. Table 4 shows a monotonic improvement from 37.61 at $r = 0\%$ to 40.38 at $r = 100\%$ (+2.77 over GRPO), with the largest gains on harder benchmarks such as AIME25 and MATH500. This suggests that recovering verifier-passing trajectories for all-zero cases is the main driver of ARPO's gains.

5 Conclusion

We identify a bottleneck in group-relative RLVR: hard instances whose rollout groups contain no verified success under binary rewards, causing normalized advantages to collapse and yielding near-zero learning signal. To address this failure mode, we propose ARPO, which detects all-zero groups and uses reference answers to recover verifier-passing step-by-step trajectories from the current policy. By reinserting such recovered trajectories into otherwise unsuccessful groups, ARPO restores informative updates on the very instances that standard GRPO-style training struggles to learn from. Experiments across diverse math reasoning benchmarks and model scales show that ARPO consistently improves solution rates, especially in sparse-success regimes with high all-zero rates. Overall, the results highlight all-zero group recovery as a simple and effective direction for strengthening RLVR on hard reasoning problems.

Limitations

Although ARPO effectively mitigates the all-zero reward regime in our experiments, this study still has several limitations. First, we restrict our investigation to mathematical reasoning tasks with programmatically verifiable binary rewards and assume access to ground-truth answers during training. This setup may be difficult to directly extend to domains where verification is noisy, expensive, or only partially available, such as open-ended question answering or tasks that inherently require graded or structured feedback rather than a single binary signal. Extending ARPO to settings with weaker supervision or imperfect answer oracles would require additional design choices on how to represent and aggregate such feedback.

Second, ARPO introduces additional inference cost whenever all sampled rollouts for a given instance receive zero reward and answer-conditioned re-rollouts are triggered. Our experimental results indicate that this overhead is manageable under the benchmarks and hyperparameter configurations considered in this work, but the extra computation may become more pronounced in scenarios with stricter latency constraints, larger models, or more aggressive rollout budgets. In such regimes, it will be important to further refine scheduling strategies for answer-refined rollouts and to explore more cost-aware variants of ARPO that better balance reasoning depth and computational efficiency.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, and et al. Zhang, Zhuosheng. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2025. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of ACL 2024*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of NeurIPS 2021*.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M Ponti, and Ivan Titov. 2025. Blending supervised and reinforcement fine-tuning with prefix sampling. *arXiv preprint arXiv:2507.01679*.

Zhensheng Jin, Xinze Li, Yifan Ji, Chunyi Peng, Zhenghao Liu, Qi Shi, Yukun Yan, Shuo Wang, Furong Peng, and Ge Yu. 2025. Recut: Balancing reasoning length and accuracy in llms via stepwise trails and preference optimization. *arXiv preprint arXiv:2506.10822*.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. 2025. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. *arXiv preprint arXiv:2509.21880*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Chen Li, Nazhou Liu, and Kai Yang. 2025. Adaptive group policy optimization: Towards stable training and token-efficient reasoning. *arXiv preprint arXiv:2503.15952*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, and et al. Li, Yanhao. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and 1 others. 2025. Ngrpo: Negative-enhanced group relative policy optimization. *arXiv preprint arXiv:2509.18851*.

Yichen Ouyang, Lu Wang, Fangkai Yang, Pu Zhao, Chenghua Huang, Jianfeng Liu, Bochen Pang, Yaming Yang, Yuefeng Zhan, Hao Sun, Qingwei Lin, Saravan Rajmohan, Weiwei Deng, Dongmei Zhang, Feng Sun, and 1 others. 2025. Token-level proximal policy optimization for query generation. In *Proceedings of EMNLP 2025*, pages 31184–31198.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL-HLT 2021*, pages 2080–2094.

Jiakang Wang, Runze Liu, Fuzheng Zhang, Xiu Li, and Guorui Zhou. 2025a. Stabilizing knowledge, promoting reasoning: Dual-token constraints for rlvr. *arXiv preprint arXiv:2507.15778*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, and et al. Zhang, Zhenru. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, and et al. Miao, Ziming. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

Fang Wu and Yejin Choi. 2025. The invisible leash: Why rlvr may not escape its origin. In *2nd AI for Math Workshop@ ICML 2025*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, and et al. Lin, Junyang. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

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

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *arXiv preprint arXiv:2504.09696*.

Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. 2025a. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *arXiv preprint arXiv:2507.02841*.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. 2025b. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. *arXiv preprint arXiv:2510.19807*.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. 2025c. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*.

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

A Appendix

A.1 Detailed Prompts of ours

In this section, we present the specific prompts used in ARPO. We employ three distinct prompt templates: the standard zero-shot chain-of-thought prompt used for policy rollouts, the answer-refined prompt used during the exploration phase, and the automated judge prompt used for validating trajectory quality.

Standard Inference Prompt. For standard policy sampling and GRPO training rollouts, we append a concise instruction to the question to elicit step-by-step reasoning and a formatted final answer:

Standard Inference Instruction

```
Let's think step by step and output the final answer
↪  within \boxed{}.
```

Answer-Refined Prompt. When an all-zero reward group is detected, we intervene using the Answer-Refined Prompt. This prompt includes the ground-truth answer and explicitly instructs the model to generate a reasoning path that leads to that answer. We also explicitly forbid the use of Python to ensure the model relies on internal chain-of-thought reasoning.

Answer-Refined Exploration Prompt

```
Please provide a step-by-step reasoning for the
↪  following problem. Your reasoning should lead to
↪  the final answer provided.

Do not use python!

questions:
{question}

answer:
{answer}

YOUR STEP-BY-STEP REASONING:
Start with your thinking process and output the final
↪  answer within \boxed{}.
```

GPT-5.1 Judge Prompt. To assess the quality of the answer-refined trajectories, we employ GPT-5.1 as an automated judge. The judge verifies not only the correctness of the final answer but also the logical validity of the reasoning process, ensuring that the model does not arrive at the correct answer via incorrect logic.

Automated Judge Prompt

```
You are a math-solution judge.

[Problem]
{problem_text}

[Student_Solution]
{student_solution}

[Official_Solution]
{official_solution_text}

Your tasks:
1. Decide whether the student's final answer is
↪  mathematically correct.
2. If and only if the final answer is correct, decide
↪  whether the reasoning process is logically and
↪  computationally sound. The method can be
↪  different from the official solution, but it must
↪  be valid and free of calculation mistakes.

Important:
- First, think through the math carefully in a field
↪  called "think". In this field, write your
↪  detailed reasoning process and comparison with
↪  the official solution if available.
- After you finish thinking in "think", also output
↪  the judgment fields.
- You MUST output a single JSON object with EXACTLY
↪  the following keys:

{
  "think": "detailed reasoning process here",
  "answer_correct": true/false,
  "process_correct": true/false,
  "comment": "very short explanation (one or two
  ↪  sentences, no full solution)"
}

Rules:
- "think" should contain your full chain-of-thought
↪  and justification.
- If the final answer is wrong, set "answer_correct":
↪  false and "process_correct": false.
- If the final answer is correct and the reasoning has
↪  no logical or arithmetic errors, set both to true.
- If the final answer is correct but there is a logic
↪  error, unjustified step, or arithmetic mistake
↪  that luckily cancels out, set "answer_correct":
↪  true and "process_correct": false.
```

A.2 Experiment Details

A.2.1 Tasks and Data Details

We evaluate our method on a suite of competition-style mathematical reasoning benchmarks spanning high-school contests to olympiad-level problems. Table 5 summarizes dataset sizes and splits used for training and evaluation in our experiments.

Table 5: Dataset sizes and data splits used for training and evaluation.

Dataset Name	Test Size	Train
AIME24	30	–
AIME25	30	–
AMC23	40	–
MATH500	500	–
Minerva	272	–
OlympiadBench	674	–
MATH	5,000	7,500

AIME24 AIME24 is a competition-math benchmark consisting of 30 integer-answer problems from AIME I & II (2024). Problems require multi-step symbolic manipulation and careful case analysis. Answers are integers (typically in the range used by AIME), which enables exact-match evaluation.

AIME25 AIME25 mirrors the AIME24 setting, containing 30 integer-answer problems from AIME I & II (2025). It is used to test generalization to a newer exam year under the same format and difficulty profile.

AMC23 AMC23 is a high-school competition benchmark curated from the 2023 AMC 12 (A and B). It contains 40 problems and covers algebra, geometry, counting, and number theory with shorter statements than AIME but still requiring non-trivial reasoning. We treat the task as free-form problem solving with exact final-answer matching.

MATH500 MATH500 is a standard evaluation subset of 500 problems drawn from the MATH test set, spanning seven subjects and multiple difficulty levels. It is widely used as a stable, medium-sized benchmark for measuring step-by-step solution quality and final-answer correctness.

Minerva Minerva (Minerva-Math) is a compact competition-math benchmark used in many LLM evaluations. It contains 272 problems and emphasizes quantitative reasoning across diverse topics (e.g., algebra, calculus, and applied math). We use it as an out-of-distribution check relative to MATH-style training data.

OlympiadBench OlympiadBench (math, English, text-only subset) contains 674 olympiad-level problems with rich solution structures, including harder combinatorics, geometry, and number theory. This benchmark is designed to stress long-horizon reasoning and compositional proof-like derivations.

MATH Dataset The MATH dataset is a large-scale collection of 12,500 competition mathematics problems with step-by-step solutions. We use the standard split of 7,500 training problems for optimization/training, and the official 5,000-problem test split for evaluation when needed. Problems span seven subjects (e.g., algebra, geometry, number theory) and multiple difficulty levels.

A.2.2 Compared Methods

We compare our approach against several representative reinforcement-learning optimizers that have been widely used for verifiable-reward reasoning model post-training. These methods largely follow the "generate a group of responses, score them with verifiable rewards, and update the policy" paradigm, but differ in how they compute advantages, apply clipping, and control length/difficulty biases.

GRPO Group Relative Policy Optimization (GRPO) is a PPO-style policy optimization method that removes the need for an explicit critic/value model. Instead, it samples a group of outputs per query and estimates the baseline/advantage from group statistics of outcome rewards, reducing memory and training overhead while maintaining strong reasoning performance.

Dr. GRPO Dr. GRPO ("GRPO Done Right") is an unbiased variant designed to correct optimization biases in vanilla GRPO that can inadvertently inflate response length, especially for incorrect outputs. It modifies the policy-gradient computation/aggregation so that training is more token-efficient and avoids uncontrolled growth in response length while preserving reasoning performance.

GPG Group Policy Gradient (GPG) revisits the classic policy-gradient objective and proposes a minimalist group-based RL approach. Compared to GRPO-style surrogates, GPG directly optimizes the original RL objective and simplifies training by eliminating auxiliary components (e.g., critic and reference models) and avoiding KL constraints, while targeting competitive reasoning performance.

DAPO Decoupled Clip and Dynamic sAmpling Policy Optimization (DAPO) is a GRPO-family algorithm tailored for long chain-of-thought RL. It decouples and re-designs clipping behavior and introduces dynamic sampling to improve efficiency and stability. The method also incorporates additional system-level and loss-level techniques aimed at preventing collapse and reducing reward noise in long-generation settings.

GSPO Group Sequence Policy Optimization (GSPO) is a stability-oriented alternative that replaces token-level importance ratios with sequence-likelihood-based importance ratios and

performs sequence-level clipping/rewarding/optimization. This design is intended to reduce token-level noise and improve training stability and efficiency, especially in large-scale settings.

GRPO-LEAD GRPO-LEAD is a difficulty-aware enhancement of GRPO for mathematical reasoning. It augments GRPO with length-regularized reward shaping to encourage concise correct solutions, explicit penalties for incorrect outputs to sharpen optimization signals, and difficulty-aware advantage reweighting so that harder problems contribute stronger learning signals.

B Implementation Details

We implement our training pipeline using the VEneralizable RL (verl) framework. The policy is optimized using Group Relative Policy Optimization (GRPO).

Training Configuration For the reinforcement learning stage, we train the model for 1 epoch with a global batch size of 128. We utilize a learning rate of 1×10^{-6} for the actor network. To ensure stability, we apply gradient clipping with a threshold of 1.0. For experience incorporation, the importance sampling ratio is clipped at $\epsilon_{low} = \epsilon_{high} = 0.2$ with an extended bound $\hat{\epsilon}_{high} = 0.6$. As we employ GRPO, we do not use a reference model for KL divergence penalty in the reward (KL coefficient set to 0); instead, we maintain a small entropy coefficient of 0.001 to encourage exploration. Additionally, we set the number of ARPO exploration samples M equal to the rollout count N, and the difficulty-aware weighting parameter λ is set to 1. The training is conducted on a single node equipped with 8 GPUs, utilizing standard data parallelism and gradient checkpointing to manage memory efficiency.

Generation and Rollout During the rollout phase, we use vLLM for efficient generation. For each query, we sample $N = 8$ responses. To facilitate long-chain reasoning, we configure a maximum prompt length of 4096 tokens and a maximum completion length of 8192 tokens. Unlike the evaluation setting in some prior works, we maintain a generation temperature of 0.6 for both training rollouts and validation to encourage diverse reasoning paths. Table 6 summarizes the detailed hyperparameters used in our experiments.

Table 6: Hyperparameters used for ARPO training.

Hyperparameter	**Value**
Algorithm	GRPO
Base Framework	verl
Number of GPUs	8
Training	
Total Epochs	1
Global Batch Size	128
PPO Mini-Batch Size	64
Learning Rate	$1e-6$
Gradient Clipping	1.0
Clip Ratio (ϵ)	0.2
Extended Clip Ratio ($\hat{\epsilon}_{high}$)	0.6
KL Coefficient	0.0
Entropy Coefficient	0.001
ARPO Specifics	
ARPO Samples (M)	N (8)
Rollout	
Number of Rollouts (N)	8
Temperature	0.6
Max Prompt Length	4096
Max Response Length	8192

C Case Study: Wrong Standard Answers vs. Correct ARPO Answers

All-zero reward groups provide no learning signal because all sampled rollouts receive zero reward. (**?**) In such cases, ARPO triggers answer-refined exploration and injects a verified-success rollout back into the group. We show instances where the standard rollouts produces an incorrect final answer, while ARPO produces the correct answer.

C.1 Example 1 : Inequality Under $\forall\, q > 0$

Problem.

Problem

What are all values of p such that for every $q > 0$, we have

$$\frac{3(pq^2 + p^2q + 3q^2 + 3pq)}{p+q} > 2p^2q?$$

Express your answer in interval notation in decimal form.

Before vs. After (final answers).

One Standard Wrong Answer vs. Our ARPO Answer

GT	$[0, 3)$
Standard (wrong)	$(-1.5, 3)$
ARPO (correct)	$[0, 3)$

Before vs. After (raw outputs).

Standard rollout output (wrong; one sample)

```
To solve the inequality \(\frac{3(pq^2 +
    p^2q + 3q^2 + 3pq)}{p+q} > 2p^2q\)
    for all \(q > 0\), we simplify the
    inequality:

1. Start by simplifying the left-hand
    side:
\[
\frac{3(pq^2 + p^2q + 3q^2 + 3pq)}{p+q} =
     \frac{3pq^2 + 3p^2q + 9q^2 + 9pq}{p+
    q}
\]

2. Bring all terms to one side:
\[
\frac{3pq^2 + 3p^2q + 9q^2 + 9pq}{p+q} -
    2p^2q > 0
\]

3. Multiply both sides by \(p+q\) (noting
     we must ensure \(p+q > 0\) for all
    \(q > 0\), which implies \(p \geq 0\)
    ):
\[
3pq^2 + 3p^2q + 9q^2 + 9pq - 2p^2q(p+q) >
     0
\]

4. Expand the last term:
\[
3pq^2 + 3p^2q + 9q^2 + 9pq - 2p^3q - 2p^2
    q^2 > 0
\]

5. Group terms:
\[
(3p - 2p^2)q^2 + (3p^2 + 9p - 2p^3)q + 9q
    ^2 > 0
\]

6. For this to hold for all \(q > 0\), we
     need the quadratic in \(q\) to be
    always positive. The leading
    coefficient must be non-negative:
\[
3p - 2p^2 + 9 > 0
\]
\[
-2p^2 + 3p + 9 > 0
\]

Solve:
\[
2p^2 - 3p - 9 < 0
\]
Roots:
\[
p = \frac{3 \pm \sqrt{9 + 72}}{4} = \frac
    {3 \pm 9}{4}
\]
So \(p = 3\) or \(p = -1.5\). The
    inequality holds between the roots:
\[
-1.5 < p < 3
\]

Since we need \(p \geq 0\), we restrict:
\[
0 \leq p < 3
\]

Let's box the final answer:
\[
\boxed{(-1.5, 3)}
\]
```

ARPO output (correct)

```
To solve the inequality \(\frac{3(pq^2 +
    p^2q + 3q^2 + 3pq)}{p+q} > 2p^2q\)
    for every \(q > 0\), we need to find
    all values of \(p\) that satisfy this
     condition.

First, let's rewrite the inequality:

\[
\frac{3(pq^2 + p^2q + 3q^2 + 3pq)}{p+q} >
     2p^2q
\]

Multiply both sides by \(p+q\). Since \(q
    >0\), but \(p+q\) could change sign
    depending on \(p\), we need to be
    careful. For the inequality to hold
    for all \(q>0\), in particular for
    very large \(q\), we must have \(p+q
    >0\) for all \(q>0\). That implies \(
    p \ge 0\), because if \(p<0\), then
    for small \(q\), \(p+q\) could be
    negative.

So assume \(p \ge 0\), then \(p+q>0\) for
     all \(q>0\), and we can safely
    multiply:

\[
3(pq^2 + p^2q + 3q^2 + 3pq) > 2p^2q(p+q)
\]

Expand:

Left side:
\[
3pq^2 + 3p^2q + 9q^2 + 9pq
\]

Right side:
\[
```

```
2p^2q(p+q) = 2p^3q + 2p^2q^2
\]

Bring everything to one side:

\[
3pq^2 + 3p^2q + 9q^2 + 9pq - 2p^3q - 2p^2
    q^2 > 0
\]

Group terms in powers of \(q\):

\[
(3p - 2p^2 + 9)q^2 + (3p^2 + 9p - 2p^3)q >
     0
\]

Factor out \(q\):

\[
q\left((3p - 2p^2 + 9)q + (3p^2 + 9p - 2p
    ^3)\right) > 0
\]

Since \(q>0\), we need:

\[
(3p - 2p^2 + 9)q + (3p^2 + 9p - 2p^3) > 0
     \quad \forall q>0
\]

This is linear in \(q\). For it to be
    positive for all \(q>0\), we need the
     coefficient of \(q\) to be
    nonnegative, and if it is zero then
    the constant term must be positive.

Coefficient of \(q\):

\[
3p - 2p^2 + 9 \ge 0
\]

Rewrite:

\[
-2p^2 + 3p + 9 \ge 0
\]
\[
2p^2 - 3p - 9 \le 0
\]

Solve \(2p^2 - 3p - 9 = 0\):

Discriminant:
\[
(-3)^2 - 4(2)(-9) = 9 + 72 = 81
\]

Roots:
\[
p = \frac{3 \pm 9}{4}
\]
\[
p = 3 \quad \text{or} \quad p = -\frac
    {3}{2}
\]
```

```
So the inequality holds for:
\[
-\frac{3}{2} \le p \le 3
\]

But we already require \(p \ge 0\), so:
\[
0 \le p \le 3
\]

Now check the constant term:
\[
3p^2 + 9p - 2p^3 = p(3p + 9 - 2p^2)
\]

At \(p=3\), coefficient of \(q\) is zero
    (since it's a root). Then we need the
     constant term to be positive:

\[
3(9) + 9(3) - 2(27) = 27 + 27 - 54 = 0
\]

So at \(p=3\), the expression becomes
    \(0\), not \(>0\). Thus \(p=3\) is
    not allowed.

Therefore:

\[
0 \le p < 3
\]

Final answer: \boxed{[0,3)}
```

Analysis. The standard rollout fails to enforce the constraint $p \geq 0$, which is necessary for $p + q > 0$ to hold for all $q > 0$. More critically, it incorrectly simplifies the linear-in-q inequality $(3p - 2p^2 + 9)q + (3p^2 + 9p - 2p^3) > 0$ by misidentifying the coefficients. For this linear expression to be positive $\forall q > 0$, the coefficient of q must be non-negative ($A \geq 0$), and if it is zero, the constant term must be strictly positive ($B > 0$). ARPO correctly identifies $A = 3p - 2p^2 + 9$ and $B = 3p^2 + 9p - 2p^3$. It finds the condition $A \geq 0$ gives $p \in [-1.5, 3]$, which is restricted to $[0, 3]$ by the $p \geq 0$ requirement. Checking the boundary case $p = 3$ (where $A = 0$) shows $B = 0$, violating $B > 0$. Therefore, $p = 3$ is excluded, leading to the correct interval $\boxed{[0, 3)}$. The standard rollout's failure to check the boundary condition and enforce $p \geq 0$ results in the incorrect answer $\boxed{(-1.5,\ 3)}$.

C.2 Example: Floor/Ceiling Product with $y < 0$

Problem.

Problem

If $y < 0$, find the range of all possible values of y such that $\lceil y \rceil \cdot \lfloor y \rfloor = 110$. Express your answer using interval notation.

Before vs. After (final answers).

One Standard Wrong Answer vs. Our Fallback Answer

GT	$\boxed{(-11, -10)}$
Standard (wrong)	$\boxed{[-110, -1)}$
ARPO (correct)	$\boxed{(-11, -10)}$

Before vs. After (raw outputs).

Standard rollout output (wrong; one sample)

```
Given the equation \(\lceil y \rceil \
    cdot \lfloor y \rfloor = 110\) and \(
    y < 0\),
we denote \(\lceil y \rceil = n\) and \(\
    lfloor y \rfloor = m\), where \(n\)
    and \(m\) are integers.
Since \(y\) is negative, both \(n\) and
    \(m\) must be negative integers.
The equation becomes \(n \cdot m = 110\).

We need pairs of negative integers whose
    product is 110:
\((-1,-110), (-2,-55), (-5,-22), (-10,-11)
    \).

For each pair, we check if there exists
    \(y\) such that \(\lceil y \rceil = n
    \) and \(\lfloor y \rfloor = m\).
This would mean \(y\) lies in an interval
     like \([m,n)\).

Thus valid intervals include \([-110,-1),
     [-55,-2), [-22,-5), [-11,-10)\).
Their union gives the range of possible
    values for \(y\), so the range is \(\
    boxed{[-110,-1)}\).
```

Fallback output (correct)

```
To find the range of all possible values
    of \( y \) such that \(\lceil y\rceil
     \cdot \lfloor y\rfloor = 110\) with
    \(y<0\), we use the relationship
    between the floor and ceiling
    functions. For any real \(y\), we
    always have
\[
\lceil y\rceil-\lfloor y\rfloor \in
    \{0,1\}.
\]
If \(y\) is an integer, then \(\lceil y\
    rceil=\lfloor y\rfloor\) and the
    product would be a perfect square,
    which cannot equal \(110\). Hence \(y
    \) is not an integer, so \(\lceil y\
    rceil=\lfloor y\rfloor+1\).

Let \(\lfloor y\rfloor = n\), so \(\lceil
     y\rceil = n+1\). Then
\[
n(n+1)=110.
\]
Solve:
\[
n^2+n-110=0 \quad \Rightarrow \quad n=\
    frac{-1\pm\sqrt{441}}{2}=\frac{-1\pm
    21}{2},
\]
so \(n=10\) or \(n=-11\). Since \(y<0\),
    we take \(n=-11\). Therefore,
\[
\lfloor y\rfloor=-11,\qquad \lceil y\
    rceil=-10.
\]
This forces \(y\) to lie strictly between
     \(-11\) and \(-10\) (endpoints
    excluded because they make \(y\) an
    integer and change the product):
\[
-11<y<-10.
\]

The final answer is:
\[
\boxed{(-11,-10)}.
\]
```

Analysis. The standard rollout incorrectly treats $\lceil y \rceil$ and $\lfloor y \rfloor$ as independent integer factors of 110, leading to a large union of intervals $\boxed{[-110, -1)}$. This is invalid because for any non-integer y, the floor and ceiling must be consecutive integers, satisfying the discrete constraint $\lceil y \rceil = \lfloor y \rfloor + 1$. Since 110 is not a perfect square, y cannot be an integer. The product must be of the form $n(n+1) = 110$. Solving this quadratic yields $\lfloor y \rfloor = n = -11$ (as $y < 0$). This uniquely forces $\lceil y \rceil = -10$. Consequently, y must lie in the open interval $(-11, -10)$, as the endpoints would make y an integer. The ARPO approach correctly uses this property, resulting in the unique correct interval $\boxed{(-11, -10)}$.

D Additional Experiments and Analyses

In this appendix, we provide several supplementary experiments and analyses that further clarify the empirical behavior of ARPO. These additions are designed to address common questions about whether ARPO's gains mainly come from extra rollout compute, whether the reported improvements remain visible under higher-budget evaluation, whether ARPO also improves the upper tail of solution discovery under larger sample budgets,

Method	Rollouts per Prompt	AIME24	AIME25	AMC23	Math500	Avg.
GRPO	8	20.94	8.44	58.98	72.20	40.14
GRPO	16	21.44	9.11	57.92	73.05	40.38
ARPO	8 (+8 answer-refined)	22.08	12.29	60.33	77.20	42.98

Table 7: Compute-matched comparison between larger-rollout GRPO and ARPO on Qwen2.5-Math-7B. Simply increasing the rollout count in GRPO yields only marginal changes, whereas ARPO achieves substantially larger gains by recovering verifier-passing trajectories on all-zero groups.

Method	AIME24	AIME25	AMC23
GRPO	20.78	8.38	55.12
ARPO	21.98	12.34	60.39

Table 8: AVG64 evaluation on smaller benchmarks. ARPO maintains consistent improvements over GRPO under a higher-budget averaging protocol.

Dataset	k	Base Model	GRPO	ARPO
AIME 2025	8	17.9	24.0	25.6
	32	26.5	32.8	35.0
	128	36.6	42.5	44.2
AMC23	8	78.9	80.0	81.3
	32	89.1	89.7	90.3
	128	97.2	95.9	96.6

Table 9: High-budget pass@k evaluation. ARPO improves solution discovery not only at pass@1, but also in higher-sampling regimes.

and whether answer-refined trajectories suffer from substantial answer leakage.

D.1 Compute-Matched Comparison Against Larger GRPO Rollout Budgets

A natural question is whether ARPO mainly benefits from additional generation budget, rather than from the proposed all-zero-group recovery mechanism itself. To examine this, we compare ARPO against GRPO under a compute-matched rollout expansion on Qwen2.5-Math-7B. Specifically, we evaluate: (i) vanilla GRPO with $N = 8$ rollouts, (ii) vanilla GRPO with $N = 16$ rollouts, and (iii) ARPO with the standard $N = 8$ rollouts plus up to 8 answer-refined trajectories when all-zero groups are detected.

As shown in Table 7, increasing the rollout count in vanilla GRPO from 8 to 16 yields only a modest change in average performance, while ARPO delivers a clearly larger gain under a comparable overall generation budget. This supports the view that ARPO's improvements are not explained by brute-force rollout scaling alone. Instead, the benefit comes from converting otherwise uninformative all-zero groups into informative updates through answer-refined trajectory recovery.

D.2 Higher-Budget Averaging on Small Benchmarks

Because AIME and AMC are relatively small test sets, low-budget repeated sampling can introduce nontrivial variance into pass@1 estimates. We therefore additionally report higher-budget averaged results using AVG64 on the smaller benchmarks.

The AVG64 results in Table 8 show the same overall pattern as the main text: ARPO remains consistently better than GRPO, with especially pronounced gains on the hardest small benchmark, AIME25. This suggests that the improvement is not an artifact of a low-budget evaluation protocol.

D.3 High-Budget Pass@k Evaluation

Our main tables focus on pass@1 following the standard evaluation protocol used throughout the paper. To further assess whether ARPO improves the upper tail of solution discovery under larger test-time budgets, we also evaluate pass@k with $k \in \{8, 32, 128\}$.

Table 9 shows that ARPO improves the probability mass assigned to verifier-passing solutions beyond the single-sample regime. The gain is especially clear on AIME 2025, where ARPO consistently outperforms GRPO at all three budgets. On AMC23, the gains are smaller, which is expected given that this benchmark is easier and already close to saturation at high k.

D.4 Answer Leakage Audit

In the main text, we evaluate the logical validity of answer-refined trajectories using an automated judge. Another important concern is whether answer-refined prompting causes the model to prematurely reveal or explicitly rely on the final answer inside intermediate reasoning. To study this, we perform an explicit answer-leakage audit on the full MATH dataset.

Metric	Count	Rate
Total answer-refined samples audited	12,485	100.0%
Leakage detected	103	0.8%
Leakage detected & final answer correct	37	0.3%

Table 10: Answer-leakage audit on answer-refined trajectories generated over the full MATH dataset. Leakage is rare, and the fraction of samples that both leak and end with a correct answer is very small.

We apply the same answer-refined prompting mechanism used in training to generate trajectories for all available MATH instances, and then use a rule-based leakage detector to identify cases in which the final answer is revealed before the designated final-answer position. We further distinguish between all leakage cases and the subset of leakage cases that still end with a correct final answer.

As shown in Table 10, answer leakage is rare overall, and only a very small fraction of samples both leak and still end with a correct final answer. In practice, this matters because ARPO only injects verifier-passing trajectories back into training. Thus, many leakage-prone and low-quality trajectories are naturally filtered out before they can affect optimization.

E ARPO Training Time Overhead Analysis

This appendix provides a more careful wall-clock time analysis for ARPO. The central point is that ARPO introduces *conditional* extra computation: it is triggered only when a rollout group for a sample yields an all-zero reward outcome.

E.1 Source of additional computation

In GRPO-style training, for each prompt we sample a rollout group of size N and obtain verifier rewards $\{r_i\}_{i=1}^{N}$ (typically $r_i \in \{0, 1\}$). We define an *all-zero* group as

$$\sum_{i=1}^{N} r_i = 0. \tag{7}$$

When this happens, there is no successful trajectory in the group, which can lead to weak or degenerate learning signal in practice.

ARPO handles this case by re-generating additional rollouts for the affected sample using an answer-refined prompt. Specifically, when an all-zero group is detected, ARPO samples an additional set of M answer-conditioned trajectories, queries the verifier, and (when successful) injects at least one positive-reward trajectory into the original group by replacing one failed trajectory. Therefore, ARPO's incremental wall-clock cost mainly comes from:

- **Extra conditional generation** for up to M additional trajectories;
- **Minor conditional verification** for these trajectories;
- **Minor overhead** for filtering/selection and bookkeeping.

Crucially, the number of trajectories used for the policy update remains N (the original group size); ARPO primarily changes *which* trajectories appear in the group when an all-zero event occurs.

E.2 Time model including backward/optimization cost

A more rigorous budget should include not only rollout generation and reward evaluation, but also the time spent on computing policy losses, gradients, and optimizer steps.

Let a *training step* denote: (i) collecting rollouts and rewards for the current batch, and (ii) performing the policy update. We decompose the expected wall-clock time per step for GRPO as

$$T_{\text{GRPO}} \;\approx\; T_{\text{rollout}}(N) + T_{\text{update}}(N), \tag{8}$$

where:

- $T_{\text{rollout}}(N)$ accounts for sampling N trajectories per prompt and running the verifier (and any advantage/statistics computation);
- $T_{\text{update}}(N)$ accounts for computing log-probabilities, loss/advantage weighting, gradient backpropagation, and the optimizer step, using N trajectories per prompt.

To make the dependence explicit, one may further write

$$T_{\text{rollout}}(N) \approx N \cdot (t_{\text{gen}} + t_{\text{ver}}) + t_{\text{misc}}, \qquad T_{\text{update}}(N) \approx N \cdot (t_{\text{fwd}} + t_{\text{bwd}}) + t_{\text{opt}}, \tag{9}$$

where t_{gen} is the average per-trajectory generation time, t_{ver} is per-trajectory verification time, $t_{\text{fwd}}/t_{\text{bwd}}$ are the per-trajectory forward/backward costs for the policy objective, and t_{opt} is the optimizer overhead.

Under ARPO, only all-zero samples trigger additional rollouts. Let p_0 be the empirical all-zero rate. Then the expected step time becomes

$$T_{\text{ARPO}} \;\approx\; T_{\text{rollout}}(N) + T_{\text{update}}(N) + \mathbb{E}[T_{\text{extra}}], \tag{10}$$

with

$$\mathbb{E}[T_{\text{extra}}] \;\approx\; p_0 \cdot \Big(M \cdot (t_{\text{gen}} + t_{\text{ver}}) + t_{\text{sel}} + t_{\text{recomp}}\Big). \tag{11}$$

Here t_{sel} denotes selection/bookkeeping overhead, and t_{recomp} captures any additional computation required to incorporate the injected trajectory into training (e.g., re-computing log-probabilities for the replaced trajectory if not already available). Importantly, because the update still uses N trajectories, *the dominant backprop/optimizer cost* $T_{\text{update}}(N)$ *is largely unchanged*; ARPO mainly adds conditional rollout-generation and verification work.

Combining the above, the expected multiplicative overhead can be expressed as

$$\frac{\mathbb{E}[T_{\text{ARPO}}]}{\mathbb{E}[T_{\text{GRPO}}]} \approx 1 + \frac{p_0 \cdot \Big(M \cdot (t_{\text{gen}} + t_{\text{ver}}) + t_{\text{sel}} + t_{\text{recomp}} \Big)}{T_{\text{rollout}}(N) + T_{\text{update}}(N)}. \tag{12}$$

E.3 Empirical wall-clock comparison and interpretation

With batch size 4 on the MATH dataset under the same training configuration, GRPO takes $9\text{h}15\text{m}$, whereas ARPO takes $11\text{h}21\text{m}$. We consider this overhead acceptable.

Importantly, a *successful* ARPO trigger implies that at least one previously all-zero reward group has been replaced by a group that contains a positive-reward trajectory. Therefore, the additional time is directly associated with recovering effective learning signals from samples that would otherwise remain entirely unsuccessful under the original rollout group.