AERO: ADAPTIVE EFFICIENT ROLLOUT OPTIMIZATION FOR VERIFIABLE-REWARD REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) assigns a fixed number of rollouts to each query. With binary rewards, all-correct and all-incorrect groups have zero group-relative advantages, while mixed groups may contain more responses than are needed for an effective update. We introduce **Adaptive Efficient Rollout Optimization** (AERO), which probes queries, reallocates a shared rollout budget to difficult groups, and reduces the responses retained for training. A Bayesian posterior supplies nonzero policy scores for groups that remain homogeneous. On Qwen2.5-family models for mathematical reasoning and single-turn coding, AERO reduces estimated generation-plus-update compute by about **48%** and per-step wall-clock time by about **45%**, under the same nominal rollout cap, while maintaining comparable Pass@8 and Avg@8. We also test adaptive rollout allocation alone in multi-turn coding, without posterior scores or rejection sampling. With Qwen3.6-27B, the $(8, 6, 4)$ schedule at $b = 16$ reaches a **60.8%** observed peak pass@1 on a 51-problem SWE-bench Verified split, versus **51.0%** for the fixed-N=16 GRPO reference, with **2.80**$\times$ recorded rollout steps per hour. Different update-group targets make this component-level evidence, not a compute-matched evaluation of full AERO.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) trains language models using automatically checked answers (Shao et al., 2024; Liu et al., 2025; Yu et al., 2025). Each step generates responses, verifies their rewards, and updates the policy. Generating a response and retaining it for the update are separate decisions. GRPO assigns a fixed number of responses to every query. With binary rewards, an all-incorrect or all-correct group has zero empirical advantages and contributes no reward-driven policy-gradient term, although regularization may still contribute. More than 35% of sampled groups can be all incorrect in the runs in Appendix B. This describes observed outcomes, not unsolvable queries.

DAPO filters homogeneous groups after generation (Yu et al., 2025). Adaptive methods can instead change where generation is spent, and some also reuse earlier responses (Xiong et al., 2025; Zhang et al., 2025). Our question is how to combine query-dependent generation with a smaller update batch: which queries should receive another attempt, and which sampled responses should contribute to the update?

A mixed group with one correct and seven incorrect responses already has a success–failure contrast. Retaining fewer incorrect responses reduces update work, but can change the gradient and remove useful diversity.

Adaptive Efficient Rollout Optimization (AERO) combines three operations. A small probe identifies all-fail groups, which receive additional rollouts from a shared budget until a success or a limit is reached. A Beta posterior supplies policy scores for homogeneous groups. For low-success mixed groups, AERO retains all correct responses and downsamples incorrect responses to a $1 : k$ ratio. We analyze the reward-balance factor that motivates $k = 1$, while distinguishing it from the response-dependent gradient direction and sampling variance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Overall results. AERO's main benefit is lower compute at comparable accuracy: about 50% and 47% fewer estimated generation-plus-update FLOPs for 1.5B and 7B math models, respectively (Table 1A–B). We also test adaptive allocation alone in multi-turn coding, without posterior scores or response rejection (Section 5.4).

2 Related Work

Adaptive allocation. AdaRFT adjusts the difficulty of sampled problems using recent rewards (Shi et al., 2025). Reinforce-Ada derives adaptive inference allocation from nonlinear RL objectives, including sequential sampling (Xiong et al., 2025). AR3PO combines adaptive rollouts with reuse of correct responses and reports lower rollout cost (Zhang et al., 2025). AERO combines within-step rescue with separate rules for retaining responses and scoring homogeneous groups.

Scores and optimization. NGRPO introduces a virtual maximum-reward sample into advantage statistics and uses asymmetric clipping to train on all-incorrect groups (Nan et al., 2025). AERO instead uses a Beta posterior for homogeneous groups; both approaches change the estimator rather than recover an observed within-group contrast. GSPO uses sequence-level likelihood ratios and clipping (Zheng et al., 2025), addressing optimization stability rather than query-level rollout allocation.

Agentic and asynchronous RL. Prefix Sampling replays successful or failing trajectory prefixes to move skewed binary-reward groups toward a balanced pass rate; replayed tokens are excluded from the loss (Zhu et al., 2026). PipelineRL overlaps generation and learning with in-flight weight updates (Piché et al., 2025). APRIL over-provisions requests and continues unfinished responses in later steps (Zhou et al., 2025). These methods respectively change the continuation state, policy synchronization, and treatment of long-running requests. AERO changes generation and retention counts. Our multi-turn experiments test only the allocation component, with event-driven collection and synchronized learner updates, without a direct comparison with these systems.

3 Preliminaries

3.1 Background: GRPO Algorithm

For a query q, GRPO samples N responses from π_{old} and assigns each response o_i a reward r_i. With $\mathbf{r} = (r_1, \ldots, r_N)$, its group-relative advantage is

$$\hat{A}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r}) + \delta}, \qquad \delta > 0. \tag{1}$$

Here δ is a numerical stabilizer, distinct from the clipping parameter ϵ. A homogeneous reward group therefore has zero advantages without an undefined $0/0$ operation. All tokens in a response share this advantage. The clipped surrogate, with KL coefficient β, is

$$J(\theta) = \mathbb{E}_{q,\{o_i\}}\Bigg[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(\rho_{i,t}\hat{A}_i, \\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\Big) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\Bigg] \tag{2}$$

where $\rho_{i,t}$ is the importance sampling ratio for the t-th token of the i-th response, defined as: $\rho_{i,t} = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{old}(o_{i,t}|q,o_{i,<t})}$. Here, π_θ is the current policy being optimized, and π_{old} is the frozen policy from which the rollouts were sampled. The clipped surrogate removes the incentive to increase a positive-advantage ratio above $1+\epsilon$ or decrease a negative-advantage ratio below $1-\epsilon$. It does not impose a hard constraint that all policy ratios remain in that interval. The KL term regularizes against a reference policy π_{ref}. Appendix C.1 gives the gradient, including importance ratios and sign-dependent clipping.

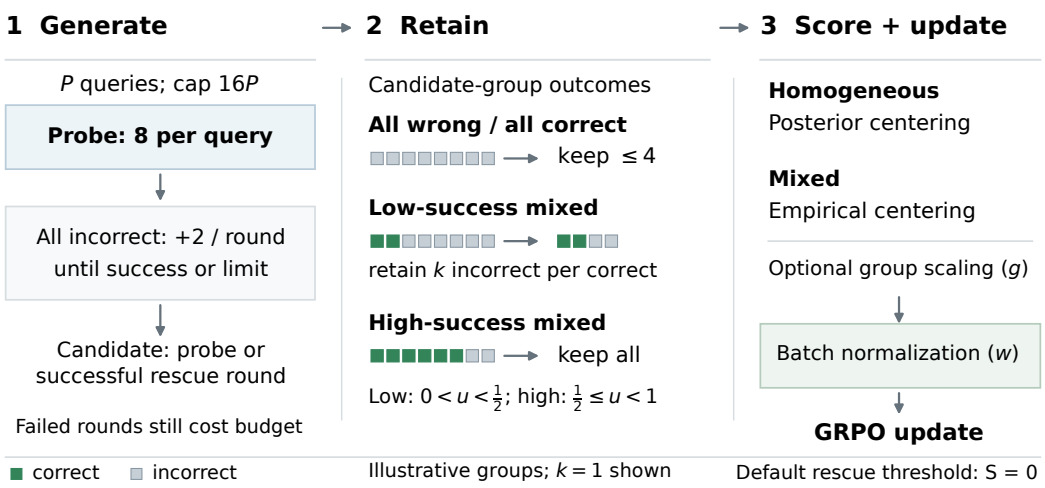


Figure 1: AERO with default rescue threshold $S = 0$. After probing and rescue, responses are retained by candidate-group outcome and scored for the update. A rescued candidate uses its first successful round, optionally combined with its probe; posterior counts include all generated outcomes. Tiles are illustrative. The reported mean retained size (≈ 4.6) is not a generation budget.

4 The AERO Method

AERO has two stages (Figure 1). Stage I probes every query with a small group of rollouts. Stage II allocates additional rollouts to the rescue set, retains a subset of responses for training, and computes policy scores. With the default rescue threshold $S = 0$, only all-incorrect probe groups receive rescue rollouts. Low-success mixed groups are downsampled, high-success mixed groups are kept, and homogeneous groups use posterior-centered scores. Algorithm 1 in Appendix A gives the complete procedure.

4.1 Stage I: Exploration & Stratification.

For P queries, let $B = n_{\text{total}}P$ be the global generation cap, where $n_{\text{total}} = 16$ is a nominal average, not a strict per-query allocation. The initial probe uses $n_{\text{explore}} = 8$ responses per query. Let u_q be its empirical success rate, and let $S \in \mathbb{N}_0$ satisfy $0 \le S < n_{\text{explore}}/2$. With $u_{\text{res}} = S/n_{\text{explore}}$, the four disjoint probe strata are

$$\begin{aligned} \mathcal{Q}_{\text{res}} &= \{q : 0 \le u_q \le u_{\text{res}}\}, \quad \mathcal{Q}_{\text{partial}} = \{q : u_{\text{res}} < u_q < 0.5\}, \\ \mathcal{Q}_{\text{high}} &= \{q : 0.5 \le u_q < 1\}, \quad \mathcal{Q}_{\text{perfect}} = \{q : u_q = 1\}. \end{aligned} \tag{3}$$

These strata describe observed probe outcomes, not oracle difficulty labels.

4.2 Stage II: Differentiated Exploitation.

The remaining allowance, $N_{\text{budget}} = (n_{\text{total}} - n_{\text{explore}})P$, is shared across rescue queries. A rescued query contributes its first successful rescue round as a candidate training pool, optionally combined with its probe. An unresolved query contributes its probe. Failed intermediate rounds still consume generation budget and update the posterior, even though they are not retained for training. Appendix E specifies the implementation switches.

4.2.1 Strategy 1: Adaptive Rescue

For the implemented $S = 0$ case, each all-incorrect query receives at most $n_{\text{extra}} = 2$ responses per rescue round after its eight-response probe. Rescue stops at the first successful round, or at the global budget, per-query limit $n_{\max}$, or round limit $K_{\max}$. The increment of two is not the total rescue budget.

Figure 5a reports successes over successive rounds, not the success rate of a single two-response increment.

4.2.2 STRATEGY 2: POSTERIOR-CENTERED SCORES

All-incorrect and all-correct candidate groups have zero empirical advantages. For these groups, AERO uses a Beta prior on the Bernoulli success probability, with $\alpha_0 = \beta_0 = 1$. After C_q successes among N_q^{gen} generated responses, including probe and rescue, the posterior parameters are $\alpha_q = \alpha_0 + C_q$ and $\beta_q = \beta_0 + N_q^{\text{gen}} - C_q$. Each generated response updates these counts once, even if it is later excluded from the training batch. The posterior mean and standard deviation are

$$\tilde{u}_q = \frac{\alpha_q}{\alpha_q + \beta_q}, \qquad \tilde{\sigma}_q = \sqrt{\frac{\alpha_q \beta_q}{(\alpha_q + \beta_q)^2(\alpha_q + \beta_q + 1)}}. \tag{4}$$

The preliminary score of a retained homogeneous-group response is

$$a_{q,i}^{\text{post}} = \frac{r_{q,i} - \tilde{u}_q}{s_q}, \qquad s_q = \begin{cases} \tilde{\sigma}_q + \delta, & g = 1, \\ 1, & g = 0, \end{cases} \quad \delta = 10^{-6}. \tag{5}$$

When enabled, the denominator uses uncertainty in the success probability, not the zero empirical reward standard deviation. Homogeneous candidate groups retain at most $h = 4$ responses; mixed groups use empirical statistics of their retained responses. A separate batch switch w normalizes these preliminary scores before optimization (Eq. (31)). The saved script matching the 1.5B AERO scores uses $g = 0, w = 1$ (Appendix E.1). The Beta prior permits count-based updates; no comparison establishes that it is the best prior. Posterior scoring changes the estimator without ranking equal-reward responses or guaranteeing a nonzero aggregate gradient. Appendix C.4 bounds preliminary scores for homogeneous histories, not gradient norms or training stability.

4.2.3 STRATEGY 3: RESPONSE RETENTION

For low-success mixed groups in $\mathcal{Q}_{\text{partial}}$, we retain all c correct responses and $m = \min(\lfloor kc \rfloor, n_q - c)$ incorrect responses from a candidate pool of size n_q, where $k \geq 1$. Selection is without replacement; the implementation supports uniform and ordered subsets (Appendix E). There is no learned "top-m most informative" ranking. Incorrect responses may dominate the sample count, but their count alone does not imply that they dominate the summed advantages or the gradient. Reward-conditioned rejection changes the training distribution even though the responses were freshly generated by π_{old}; the usual policy importance ratio does not correct this selection bias.

4.3 THEORETICAL GUIDANCE FOR REJECTION SAMPLING

Theorem C.1 separates a scalar reward-balance factor from the response-dependent gradient directions. It motivates examining $k = 1$, but does not establish a generally optimal downsampling ratio. For fixed positive and negative counts c, m, uniform response length, inactive clipping, no KL term or batch normalization, and population reward standard deviation, the sampled group gradient has the identity

$$\widehat{G}_m = \frac{\sqrt{cm}}{c + m}(\bar{v}_C - \bar{v}_F), \qquad v_i = \frac{1}{L} \sum_t \rho_{i,t} \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}). \tag{6}$$

The squared scalar factor is largest at $m = c$. The response vectors also change with the selected subset. Appendix C.5 shows the fixed-pool mean and added variance for uniform subsampling. This auxiliary analysis does not cover ordered selection or establish an optimal ratio for the batch-normalized implementation.

4.4 COMPUTATION COST SAVING ACROSS ALL STRATEGIES

The reported retained count is $n_{\text{training}} \approx 4.6$ responses per query, versus 16 for GRPO. The generated count includes every probe and rescue response, including discarded ones. Smaller retained batches reduce update work, while sequential rescue can add generation latency. Net savings depend on response lengths, utilization, and the relative costs of generation and training.

5 EXPERIMENTS

We report compute, wall-clock time, and held-out accuracy for full AERO on single-turn tasks, then test adaptive allocation alone in multi-turn coding agents.

5.1 Experimental Setup

Core Model & Hardware. The single-turn experiments in this section use the **Qwen2.5-Math-1.5B**, **Qwen2.5-1.5B-Base**, **Qwen2.5-Math-7B**, and **Qwen2.5-7B-Instruct-1M** models. Training for the 1.5B models is performed on **2× NVIDIA A10 GPUs**, while the 7B models are trained on **4× NVIDIA A100 GPUs**.

Datasets. For math, OpenR1-AERO contains 40k problems curated from the `OpenR1-Math-220k` dataset (Open R1, 2025). Validation and testing are conducted on held-out sets covering five reasoning sub-domains: **AIME**, **AMC**, **MATH**, **Minerva**, and **Olympiad** for broader evaluation. For the **Code** domain, training uses the dataset of 1.4k coding problems introduced by Cui et al. (2025). Validation and testing use 0.5k held-out sets from **APPS** (Hendrycks et al., 2021), **CodeContests** (Li et al., 2022), and **TACO** (Li et al., 2023).

Training Protocol. For each method, we select the single best-performing checkpoint based on the highest average score achieved on the validation set. This selected checkpoint is then used for a final, one-time evaluation on the held-out test set to report our main results. For Pass@8 and Avg@8, every evaluated policy generates eight fresh responses for each held-out problem under the same evaluation protocol. Training-time allocation, filtering, and retained group size do not determine the number of evaluation samples.

Baselines and Method Configuration. We compare AERO with GRPO and DAPO using the same nominal per-step generation cap. This controls the response-count allowance, not necessarily generated tokens, retained tokens, or wall-clock cost. (1) **GRPO (Main Baseline):** The standard GRPO algorithm is run with a fixed budget of $n = 16$ rollouts for every problem. (2) **DAPO:** We compare against the DAPO algorithm, which incorporates techniques like clipping and adaptive sampling. It also uses a total generation budget of $n = 16$. (3) **AERO (Ours):** The batch-level generation cap is $16P$ for P queries. The probe uses $8P$ responses, leaving a shared rescue allowance of at most $8P$, subject to per-query and round limits. We use a downsampling ratio of $k = 1$ for rejection sampling and a weak Bayesian prior of Beta$(\alpha_0 = 1, \beta_0 = 1)$ for zero-advantage cases.

5.2 Main Results

Compute and Time Performance. Table 1A–B reports math results; Table 2 gives per-benchmark accuracy. Figure 6 separates generation and update costs.

Step time falls from 34.3 to 19.1 minutes for 1.5B and from 21.5 to 10.8 minutes for 7B; model-update time falls by 61% and 68%, respectively. The total FLOP estimates fall by about half. Rollout estimates use the nominal generation allowance, not matched realized token counts; the mean retained group size alone is not a compute budget.

Training Performance. The recorded training rewards are broadly comparable (Figure 7a). Group-composition plots are retained in Appendix B; they are descriptive, not independent evidence of improved stability.

Testing Performance. AERO reaches Avg@8 of 0.3452 and 0.4800, and Pass@8 of 0.6121 and 0.7000, for 1.5B and 7B. The small differences from GRPO and DAPO do not establish uniform statistical superiority.

5.2.1 Additional Baselines

To better compare the compute and performance trade-off with AERO, we evaluate GRPO under Reduced-Budget Rollouts. We uniformly reduce the rollout budget per query from the default $N = 16$ to smaller values ($N/2$,$0.32N$,$N/4$,$N/8$).

Table 1A summarizes the reduced-budget controls; Appendix D.3.1 reports other model configurations. Figure 8 separates rollout and update costs.

The $N = 8$ control uses 1863 PFLOPs, close to AERO's 1811, with Avg@8 of 0.3211 versus 0.3452. The approximately five-response control ($0.32N$ in the original report) has similar step time (20.1 versus 19.1 minutes), but lower estimated compute (1133 versus 1811 PFLOPs) and Avg@8 (0.3184). Neither comparison simultaneously matches generation tokens, update tokens, and query composition. They show a cost–accuracy trade-off, but do not distinguish adaptive allocation from

Table 1: Efficiency and accuracy across tasks. (A–D) Time and estimated generation-plus-update FLOPs are per step; evaluation uses eight fresh responses per problem. Parentheses in (A) give reported p-values. (E) Multi-turn results for adaptive allocation only; ratios and peak differences use the $N = 16$ GRPO row. Unequal group targets and run lengths limit this comparison.

(A) Math: Qwen2.5-Math-1.5B; default $N = 16$

Method	Time (min)	FLOPs (PF)	Avg@8	Pass@8
GRPO	34.3	3627	0.3401 ($= 0.28$)	0.6090
DAPO	26.1	2722	0.3405 ($= 0.31$)	0.6100
AERO	19.1	1811	**0.3452**	**0.6121**
GRPO $N/2$	23.7	1863	0.3211 ($= 0.004$)	0.5954
GRPO $0.32N$	20.1	1133	0.3184 ($= 0.002$)	0.5891
GRPO $N/4$	16.3	906	0.3133 ($= 0.001$)	0.5870
GRPO $N/8$	8.9	453	0.2820 ($< 10^{-4}$)	0.5791
Base model	—	—	0.1617 ($< 10^{-6}$)	0.5050

(B) Math: Qwen2.5-Math-7B

Method	Time (min)	FLOPs (PF)	Avg@8	Pass@8
GRPO	21.5	17334	0.4690	0.6860
DAPO	15.8	13743	0.4750	0.6910
AERO	10.8	9181	**0.4800**	**0.7000**

(C) Single-turn coding: Qwen2.5-7B-Instruct-1M

			Avg@8				
Method	Time (min)	FLOPs (PF)	TACO	CodeContests	APPS	Total	p-value
AERO	6.95	2295	**0.179**	**0.321**	**0.337**	**0.264**	—
GRPO	11.66	4488	0.157	0.296	0.298	0.238	< 0.005
DAPO	7.75	2941	0.162	0.313	0.310	0.252	$= 0.034$
Base model	—	—	0.094	0.195	0.188	0.149	$< 10^{-5}$

(D) Component comparisons: Qwen2.5-Math-1.5B

Method	Time (min)	FLOPs (PF)	Avg@8	Pass@8
AERO	19.1	1811	0.345	0.612
w/o Rejection Sampling	22.9	2099	0.340	0.606
w/o Advantage Filtering	26.8	2732	0.329	0.598
Separate Bayesian comparison: batch size 128				
GRPO + Bayesian Prior	20.8	2186	0.322	0.594
GRPO	21.9	2319	0.318	0.580

(E) SWE-bench Verified: Qwen3.6-27B, $b = 16$; allocation only

Setting	Rollout steps	Hours	Steps/h	Ratio	Peak pass@1	Δ (pp)
GRPO $N = 16$	62	71.5	0.87	1.00×	0.510 (26/51)	—
GRPO $N = 8$	109	84.2	1.29	1.48×	0.549 (28/51)	+3.9
Allocation $(8, 6, 4)$	199	81.4	2.44	2.80×	**0.608 (31/51)**	+9.8
Allocation $(8, 4, 4)$	109	60.0	1.82	2.09×	0.569 (29/51)	+5.9
Allocation $(8, 8, 8)$	149	89.4	1.67	1.92×	**0.608 (31/51)**	+9.8

Table 2: Per-benchmark held-out accuracy for Qwen2.5-Math models. Each policy generates eight fresh responses per problem. Parentheses give the reported p-values for overall Avg@8; per-step time and FLOPs are in Table 1.

Avg@8							
Model	Method	AIME	AMC	MATH	Minerva	Olympiad	Total
1.5B	AERO	0.107	0.323	0.578	0.171	0.256	**0.3452**
	GRPO	0.083	0.326	0.571	0.173	0.251	0.3401 ($= 0.28$)
	DAPO	0.101	0.330	0.561	0.171	0.257	0.3405 ($= 0.31$)
	Base	0.025	0.192	0.322	0.085	0.129	0.1617 ($< 10^{-6}$)
7B	AERO	0.214	0.552	0.719	0.289	0.382	**0.4800**
	GRPO	0.208	0.543	0.714	0.267	0.372	0.4690 ($= 0.05$)
	DAPO	0.208	0.533	0.720	0.281	0.376	0.4750 ($= 0.29$)
	Base	0.138	0.358	0.539	0.179	0.196	0.3110 ($< 10^{-6}$)
Pass@8							
Model	Method	AIME	AMC	MATH	Minerva	Olympiad	Total
1.5B	AERO	0.286	0.660	0.863	0.409	0.517	**0.6121**
	GRPO	0.238	0.604	0.856	0.421	0.519	0.6090
	DAPO	0.333	0.642	0.843	0.409	0.526	0.6100
	Base	0.200	0.700	0.778	0.345	0.399	0.5050
7B	AERO	0.476	0.811	0.905	0.503	0.626	**0.7000**
	GRPO	0.476	0.721	0.902	0.491	0.607	0.6860
	DAPO	0.476	0.736	0.905	0.491	0.619	0.6910
	Base	0.400	0.700	0.876	0.434	0.510	0.6220

all possible curriculum effects. A random-retention or allocation-shuffled control remains needed for that attribution.

5.2.2 Performance on Code Generalization Tasks.

On single-turn coding (Table 1C), AERO records Avg@8 of 0.264, compared with 0.238 for GRPO and 0.252 for DAPO, with the lowest step time (6.95 minutes) and estimated compute (2295 PFLOPs) among the three methods.

5.3 Component and Group Diagnostics

Table 1D reports the available ablations, including a separate Bayesian comparison at batch size 128. These are not a fully factorial component study.

Disabling rejection sampling increases step time by 19% and FLOPs by 16%. The cost sweep in Appendix D.3.2 finds 7%, 13%, and 16% more training time at $k = 2, 3, 4$ than at $k = 1$; it does not establish the accuracy-optimal ratio.

The recorded "w/o Advantage Filtering" row has higher cost and lower accuracy than full AERO. Its intervention is not fully specified by the available result label, so it cannot isolate the effect of a standalone post-hoc filtering control.

Bayesian scores. At batch size 128, GRPO with posterior-centered scores achieves Avg@8 of 0.322 versus 0.318 and Pass@8 of 0.594 versus 0.580.

Its recorded gradient norms are about $1.5\times$ larger (Appendix D.3.3). This measures magnitude, not variance, alignment with an unfiltered gradient, or stability across independent runs.

Group composition. For 1.5B, the all-incorrect fraction is 0.261 for AERO, 0.336 for GRPO, and 0.345 for DAPO (Figure 5b): 7.5 and 8.4 percentage points lower. This full-method diagnostic neither isolates rescue nor determines gradient direction.

Downsampling high-success mixed groups ($0.5 \leq u < 1$) reduces accuracy with little time saving in the additional comparison in Appendix D.3.4.

5.4 Multi-Turn Coding: Adaptive Allocation Only

We evaluate AERO's adaptive rollout allocation alone on multi-turn coding, without posterior-centered scores or mixed-group rejection sampling. Completed groups are processed immediately and freed capacity is refilled. The behavior policy stays fixed during collection; the learner updates after collection and draining. Experiments use Qwen3.6-27B, SGLang, and Megatron on two nodes with 16 H200 GPUs, with 449 SWE-bench Verified training tasks and a 51-task evaluation split.

At $b = 16$ fresh queries per wave (Table 1E; Figure 2), the $(8, 6, 4)$ schedule uses 8 initial rollouts and optional shots of 6 and 4. It records 2.44 rollout steps/h and a peak 31/51 solved, versus 0.87 steps/h and 26/51 for GRPO $N = 16$: 2.80× the step rate and a 9.8-percentage-point higher observed peak pass@1. The allocation variant targets four mixed groups per step, versus eight for this reference, so steps/h does not measure matched training-group or token throughput. Peaks come from single runs of unequal duration.

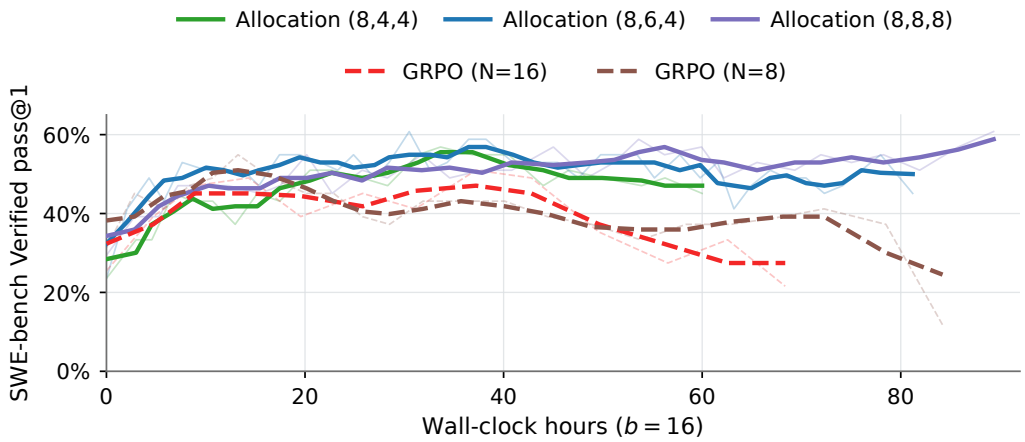


Figure 2: Learning curves for Table 1E on the 51-task SWE-bench Verified split ($b = 16$). Faint lines show all evaluations; bold lines are centered three-point means. Time starts at each run's first evaluation (Appendix C.7).

6 Conclusion

AERO combines adaptive rollout allocation, response downsampling, and posterior-centered scores for homogeneous groups. On Qwen2.5-Math, it uses about half the estimated FLOPs and reduces step time at comparable Pass@8 and Avg@8, with similar savings in single-turn coding. Adaptive allocation alone records 2.80× the reference rollout steps/h and a 9.8-point higher observed peak pass@1 in multi-turn coding at $b = 16$. Different update-group targets limit this comparison.

7 Limitations

Full AERO is tested on 1.5B/7B binary-reward single-turn tasks; the 27B multi-turn results test only adaptive allocation. Continuous rewards, process supervision, open-ended generation, and learned reward models need separate statistical treatment and experiments. Matched comparisons with newer adaptive methods and PipelineRL/APRIL are missing.

The runs lack multi-seed uncertainty and factorial ablations. Posterior scoring and reward-conditioned retention change the estimator; larger scores need not improve updates. Gradient-variance, prior-sensitivity, and token-matched controls remain needed. Latency depends on generation, verification, updates, and stragglers; sequential rescue can be slower when generation dominates. The caps and retention ratio are not proven optimal.

IMPACT STATEMENT

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

AI USE STATEMENT

OpenAI Codex assisted with literature retrieval, bibliography organization, prose and LaTeX editing, reviewer-response analysis, and the mathematical derivations and numerical checks added in this revision. The training results reported here come from existing experiment records; the assistant did not generate new training measurements. The authors remain responsible for verifying the final text, mathematics, citations, and experimental claims.

REPRODUCIBILITY STATEMENT

Algorithm 1 in Appendix A specifies generation, candidate-pool construction, retention, and scoring. Appendix C gives the assumptions and proofs; Appendices E–G describe configuration, data, and rewards. Section 5.4 separates the allocation-only experiments from full AERO and identifies the differing update-group targets. Run-level configuration provenance must be retained when reproducing either implementation.

REFERENCES

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. *arXiv preprint arXiv:2105.09938*, 2021.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:33125–33139, 2022.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yujia Li, David Choi, Junyoung Chung, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

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

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

Open R1. OpenR1-Math-220k. Hugging Face dataset, 2025. URL `https://huggingface.co/datasets/open-r1/OpenR1-Math-220k`.

Alexandre Piché, Ehsan Kamalloo, Rafael Pardinas, Xiaoyin Chen, and Dzmitry Bahdanau. PipelineRL: Faster on-policy reinforcement learning for long sequence generation. *arXiv preprint arXiv:2509.19128*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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*, pp. 1279–1297, 2025.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-Ada: An adaptive sampling framework under non-linear RL objectives. *arXiv preprint arXiv:2510.04996*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, 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, 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. *arXiv preprint arXiv:2503.14476*, 2025.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. Improving sampling efficiency in RLVR through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025.

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

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, Chenyang Zhao, Jin Pan, Xiaodong Yu, Ze Wang, Kangrui Du, Jialian Wu, Ximeng Sun, Jiang Liu, Qiaolin Yu, Hao Chen, Zicheng Liu, and Emad Barsoum. APRIL: Active partial rollouts in reinforcement learning to tame long-tail generation. *arXiv preprint arXiv:2509.18521*, 2025.

Tianshu Zhu, Wenyu Zhang, Xiaoying Zuo, Lun Tian, Haotian Zhao, Yucheng Zeng, Jingnan Gu, Daxiang Dong, Jianmin Wu, Dawei Yin, and Dou Shen. Rollout pass-rate control: Steering binary-reward RL toward its most informative regime. *arXiv preprint arXiv:2605.05112*, 2026.

A FULL AERO ALGORITHM

Algorithm 1: AERO: generation, curation, and policy update ($S = 0$)

Input: P queries; frozen rollout policy π_{old}; generation cap $B = n_{\text{total}}P$; probe size n_{explore}; rescue increment n_{extra}; limits $n_{\max}, K_{\max}$; ratio k; homogeneous retention count h; combine-probe switch η; subset rule sel; group/batch normalization switches g, w.

Output: Curated batch $\mathcal{D}_{\text{train}}$ and updated policy.

Initialize fresh query identifiers, Beta priors, $\mathcal{D}_{\text{train}} \leftarrow \varnothing$, and $U \leftarrow 0$;

Generate probe P_q of size n_{explore} for each query; set $N_q^{\text{gen}} \leftarrow |P_q|$ and $U \leftarrow \sum_q |P_q|$; score and update each posterior once per response;

Stratify probe groups into $\mathcal{Q}_{\text{res}}, \mathcal{Q}_{\text{partial}}, \mathcal{Q}_{\text{high}}, \mathcal{Q}_{\text{perfect}}$ as defined in Section 4.1;

for *each rescue round, up to* $K_{\max}$ **do**

  Build rescue requests in the controller's query order, at most $\min(n_{\text{extra}}, n_{\max} - N_q^{\text{gen}})$ per unresolved query; truncate the request list to $B - U$;

  if *the request list is empty* **then**

    stop rescue;

  end

  Generate and score the requests; increment U and N_q^{gen} by actual completions; update each posterior once per new response;

  If query q has a success, save this round as H_q and remove q from rescue;

end

for *each query* q **do**

  Set $T_q = P_q$ if not rescued; otherwise set $T_q = H_q$ (or $P_q \cup H_q$ if $\eta = 1$). Let $n_q = |T_q|$ and $c_q = \sum_{i \in T_q} r_{q,i}$;

  if $c_q = 0$ *or* $c_q = n_q$ **then**

    Retain $\min(h, n_q)$ responses using sel; compute preliminary scores using Eq. (5) with switch g;

  else

    if $0 < c_q < n_q/2$ **then**

      Retain all c_q correct responses and $m_q = \min(\lfloor kc_q \rfloor, n_q - c_q)$ incorrect responses using sel without replacement;

    else

      Retain the candidate pool;

    end

    Center rewards within the retained mixed group; divide by its empirical standard deviation plus δ if $g = 1$;

  end

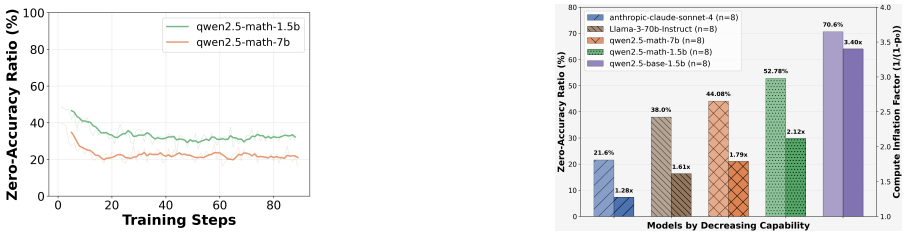
  Append retained responses and scores to $\mathcal{D}_{\text{train}}$;

end

Collate and pad the update batch for distributed execution;

If $w = 1$, normalize preliminary scores over that batch as in Eq. (31);

Apply the clipped GRPO objective to $\mathcal{D}_{\text{train}}$, then synchronize the rollout policy for the next step;



(a) (b)

Figure 3: Zero-accuracy problems in GRPO-style training and inference: (a) zero-accuracy problem ratio during GRPO training; (b) Proportion of zero-accuracy problems during rollout inference across model sizes, averaged over five benchmarks ($n = 8$). Light bars show the zero-accuracy ratio p_0 (%, left y-axis). Dark bars show $1/(1-p_0)$ (right y-axis), the expected draws per surviving group under independent sampling with fixed all-incorrect probability p_0. This excludes all-correct groups and is not a hardware-speedup estimate. Text annotations (e.g., $1.28\times$) report the inflation factor.

B Additional Diagnostics and Detailed Results

B.1 Observed All-Incorrect Groups

B.2 Unclipped Gradient

Starting from Eq. (2), assume that $1-\epsilon < \rho_{i,t} < 1+\epsilon$ for every token, so neither clipping branch is active. Treating sampled rewards and advantages as fixed during the update, the gradient simplifies to

$$\nabla_\theta J(\theta) = \mathbb{E}_{q,\{o_i\}}\Bigg[\frac{1}{N}\sum_{i=1}^{N}\frac{\hat{A}_i}{|o_i|}\sum_{t=1}^{|o_i|} \rho_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t}\,|\,q, o_{i,<t})\Bigg] - \beta\,\nabla_\theta D_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}}). \tag{7}$$

The proof can be found in Appendix C.1.

Fixed-size sampling can spend substantial compute on all-incorrect or all-correct groups. Their zero advantages remove the reward-driven policy-gradient contribution, although regularization terms may remain. This motivates adapting both the generated and retained group sizes.

B.3 The Impact of Zero-Advantage Problems on GRPO

We call a query whose sampled responses are all incorrect a **zero-accuracy problem**. This is an observed status of a sampled group, not a claim that the query's true success probability is zero.

When all rollouts for a given problem fail (i.e., the rewards $r_1, \ldots, r_N = 0$), the empirical success rate $u = \text{mean}(\mathbf{r}) = 0$, leading to a complete collapse of the advantage,

$$\hat{A}_i = \frac{r_i - u}{\text{std}(\mathbf{r}) + \delta} = \frac{0-0}{0+\delta} = 0. \tag{8}$$

Figure 4 pools 82 observations from steps 1–41 of two GRPO configurations: GRPO-1 uses $N = 16$ responses per query, and GRPO-2 uses $N = 4$. We consider two quantities: (1) **zero-accuracy problem ratio**, x_t^{zero}: the fraction of problems whose rollouts are all incorrect:

$$x_t^{\text{zero}} = \frac{\sum_{p=1}^{P}\mathbb{I}(\text{All } r_i = 0 \text{ for problem } p)}{P}. \tag{9}$$

where P is the number of problems in the batch and $\mathbb{I}(\cdot)$ is the indicator function.

(2) **Centered training reward**: let $y_{t,i}$ be the mean training reward recorded for configuration i at step t. The plotted value is

$$\tilde{y}_{t,i} = y_{t,i} - \frac{y_{t,1} + y_{t,2}}{2}. \tag{10}$$

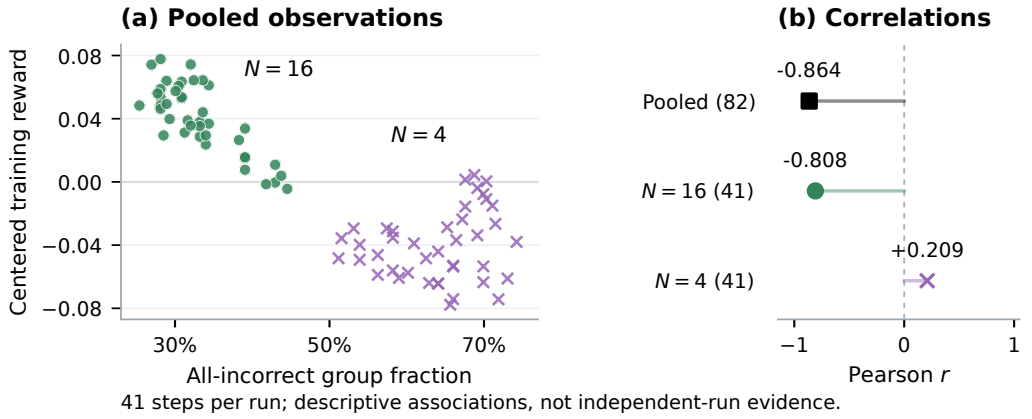


Figure 4: All-incorrect groups and centered training reward. (a) The same 82 observations from two GRPO runs, without smoothing. (b) Pearson correlations for the pooled data and each run separately; parentheses give observation counts, not independent seeds. Group-size differences and separated clusters can contribute to the pooled trend. The within-run correlations differ, and successive checkpoints are not independent observations.

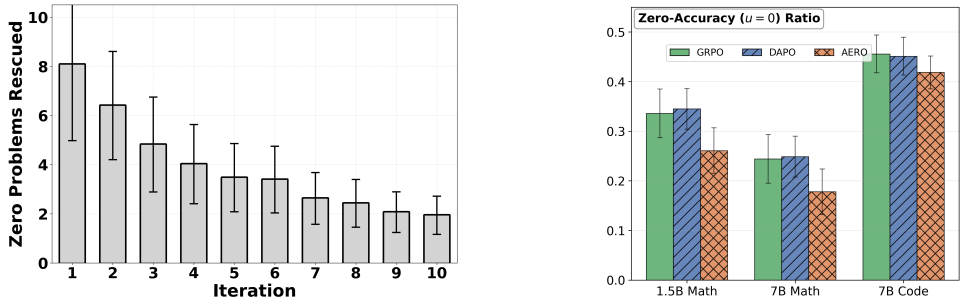
This is training reward relative to the two-configuration step mean, not a held-out test score or a ratio-to-maximum normalization.

The pooled Pearson correlation is $r = -0.864$. This association can reflect differences in group size as well as changes during training. It does not establish that reducing all-incorrect groups causes improved held-out performance. We do not interpret the pooled checkpoints as independent-run evidence of statistical significance.

Recomputing correlations separately on the same 41 steps gives $r = -0.808$ for GRPO $N = 16$ and $r = +0.209$ for GRPO $N = 4$, using the plotted centered reward. Using each run's uncentered training reward gives -0.920 and -0.974, respectively. These are different statistical quantities: centering subtracts the two-run mean at each step and couples the two series. The pooled plot cannot support a universal within-run linear relation, nor does it measure generalization. We report these coefficients descriptively, without checkpoint-level significance tests.

All-correct groups also have zero empirical advantages. Theorem C.1 analyzes mixed groups under simplifying assumptions; the homogeneous case follows directly from the stabilized advantage definition above.

B.4 Additional Single-Turn Results



(a) Rescued queries by round. (b) All-incorrect groups by model.

Figure 5: Single-turn group diagnostics: (a) rescued queries over successive rescue rounds; (b) mean all-incorrect-group ratio over training steps. Original error bars are retained; no independent-seed confidence intervals are reported.

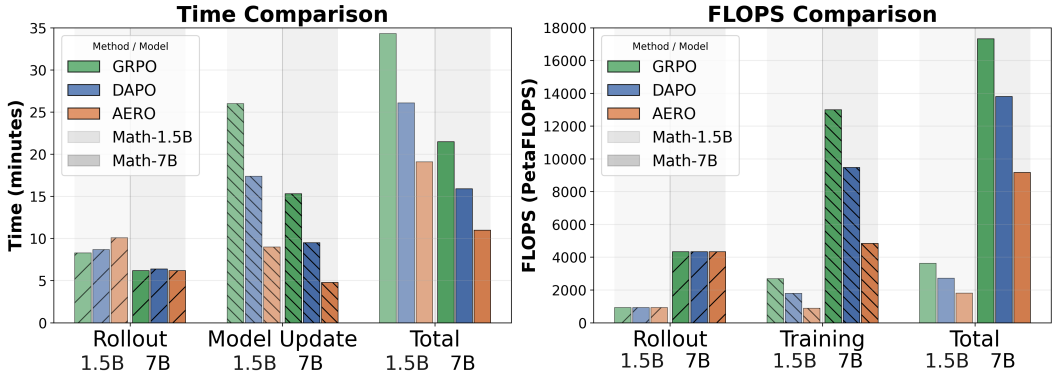
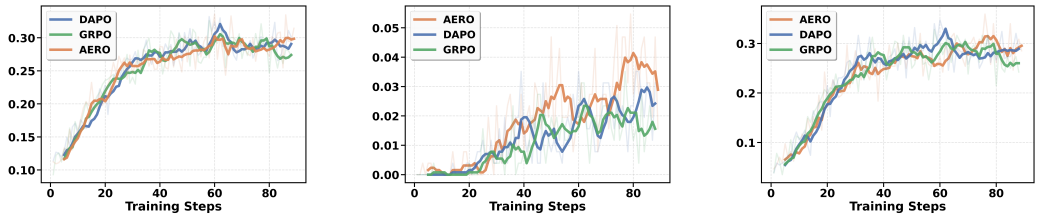


Figure 6: Per-step (a) wall-clock time and (b) estimated FLOPs, decomposed into rollout generation, model update, and their total.



(a) Training reward. (b) All-correct groups. (c) High-success mixed groups.

Figure 7: Training performance on the 1.5B model.

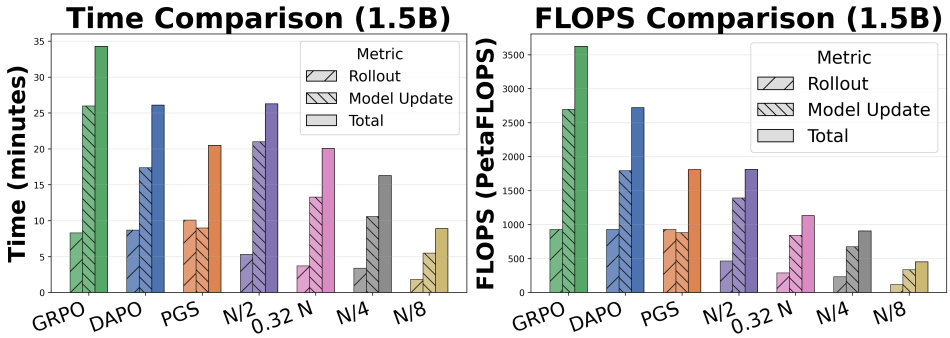


Figure 8: Per-step time and estimated FLOPs across baselines. The legacy label PGS denotes AERO.

C Supplementary Proofs

Theorem C.1 (GRPO Gradient Norm Bound)**.** *Fix positive counts c, m and let $n = c + m$. A retained group has c correct responses in F_C and m incorrect responses in F_S. Assume uniform response length L, inactive clipping, zero KL penalty, no batch normalization, and population reward standard deviation, omitting the numerical stabilizer for this analysis. Conditional on these counts, define*

$$G(\theta) = \mathbb{E}_{q,\{o_i\}|c,m}\left[\frac{1}{nL}\sum_{i=1}^{n}\sum_{t=1}^{L}\hat{A}_i\,\rho_{i,t}\,\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right], \tag{11}$$

where $\rho_{i,t}$ is the token importance ratio. Then

$$\|G(\theta)\|^2 \le \frac{c\,m}{(c+m)^2}\,\mathbb{E}\big[\|\bar{v}_C - \bar{v}_F\|^2 \mid c, m\big],$$

where $\bar{v}_C = \frac{1}{c}\sum_{j\in F_C} v_j$, $\bar{v}_F = \frac{1}{m}\sum_{j\in F_S} v_j$, and $v_j = \frac{1}{L}\sum_{t=1}^{L}\rho_{j,t}\nabla_\theta \log \pi_\theta(o_{j,t} \mid q, o_{j,<t})$.

The proof is in Appendix C.2. The importance ratios remain inside v_j; they equal one only when $\pi_\theta = \pi_{\text{old}}$. The directional term can change when the retained subset changes.

Lemma C.2 (Maximization of the Balance Factor)**.** *For $c, m > 0$, the scalar factor $cm/(c+m)^2$ is at most $1/4$, with equality if and only if $m = c$.*

Setting $k = 1$ gives this balanced count when enough incorrect responses are available. Appendix C.3 proves the scalar claim. Maximizing one factor in an upper bound does not establish that the bound, the actual gradient norm, or learning performance is maximized. The bound alone says nothing about gradient variance; the additional fixed-pool calculation in Appendix C.5 addresses only the randomness from uniform negative subsampling.

C.1 Policy-Gradient Derivation

Proof. The clipped surrogate must retain the signed advantage inside both arguments of the minimum:

$$J(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} f_\epsilon(\rho_{i,t}, \hat{A}_i)\right] - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}),$$
$$f_\epsilon(\rho, A) = \min\{\,\rho A,\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\,\}. \tag{12}$$

Moving A outside the minimum is invalid when $A < 0$. With the sampling distribution, π_{old}, and the advantages fixed,

$$\nabla_\theta \rho_{i,t} = \rho_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}). \tag{13}$$

Away from the clipping boundaries,

$$\frac{\partial f_\epsilon}{\partial \rho} = \begin{cases} A, & A > 0 \text{ and } \rho < 1+\epsilon, \\ A, & A < 0 \text{ and } \rho > 1-\epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Boundary subgradients may be chosen by the implementation. Define

$$U_i = \{t : (\hat{A}_i > 0 \wedge \rho_{i,t} < 1+\epsilon) \ \vee\ (\hat{A}_i < 0 \wedge \rho_{i,t} > 1-\epsilon)\}.$$

The gradient is therefore

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\hat{A}_i}{|o_i|}\sum_{t\in U_i}\rho_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right] - \beta\nabla_\theta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}). \tag{15}$$

If every ratio lies strictly between $1-\epsilon$ and $1+\epsilon$, all tokens with nonzero advantage are active. Thus

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\hat{A}_i}{|o_i|}\sum_{t=1}^{|o_i|}\rho_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right] - \beta\nabla_\theta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \tag{16}$$

which gives Eq. (7). Setting $\epsilon = 0$ does not disable clipping; it collapses the clipping interval. □

C.2 Proof of Theorem C.1

Under the assumptions of Theorem C.1, let $n = c + m$ with fixed $c, m > 0$. Write the conditional expected group gradient as

$$G(\theta) = \mathbb{E}_{q,\{o_i\}|c,m}\left[\frac{1}{nL}\sum_{i=1}^{n}\sum_{t=1}^{L}\hat{A}_i\rho_{i,t}\,\nabla_\theta \log \pi_\theta(o_{i,t}\,|\,q, o_{i,<t})\right]. \tag{17}$$

The expectation is conditional on the counts; otherwise the count-dependent coefficient must remain inside the expectation.

Proof. Keep both the token importance ratio and the length normalization in the response vector:

$$v_i = \frac{1}{L}\sum_{t=1}^{L}\rho_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}).$$

For a fixed sampled group, the reward-gradient term is

$$\hat{G}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\hat{A}_i v_i. \tag{18}$$

With $\mu = c/n$ and population variance $\sigma^2 = cm/n^2$, the standardized binary advantages are $a_+ = \sqrt{m/c}$ and $a_- = -\sqrt{c/m}$. Defining $\bar{v}_C = c^{-1}\sum_{i\in F_C} v_i$ and $\bar{v}_F = m^{-1}\sum_{i\in F_S} v_i$ gives the exact identity

$$\hat{G} = \frac{ca_+\bar{v}_C + ma_-\bar{v}_F}{n} = \frac{\sqrt{cm}}{c+m}(\bar{v}_C - \bar{v}_F).$$

Consequently, for fixed counts and finite second moments,

$$\begin{aligned}\|G\|^2 &= \frac{cm}{(c+m)^2}\big\|\mathbb{E}[\bar{v}_C - \bar{v}_F \mid c, m]\big\|^2 \\ &\le \frac{cm}{(c+m)^2}\mathbb{E}[\|\bar{v}_C - \bar{v}_F\|^2 \mid c, m],\end{aligned}$$

by Jensen's inequality. The homogeneous cases $c = 0$ or $m = 0$ are excluded from this standardized mixed-group derivation; their stabilized empirical advantages are zero by definition.

□

C.3 Proof of Lemma C.2

Lemma C.3 (Maximization of the Balance Factor)**.** *For fixed $c > 0$, $z(m) = cm/(c+m)^2$ is maximized over $m > 0$ if and only if $m = c$.*

Proof. The derivative is $z'(m) = c(c-m)/(c+m)^3$. It is positive for $m < c$, zero at $m = c$, and negative for $m > c$. Thus $z(c) = 1/4$ is the unique maximum. The same conclusion holds for positive integer counts. This proves only the scalar claim: the directional term in Theorem C.1 also depends on the selected responses, so neither the full bound nor the actual gradient norm is necessarily maximized at $k = 1$. □

C.4 Posterior Scores for Homogeneous Histories

For $n \ge 1$ observations that are *all incorrect*, the $\mathrm{Beta}(1,1)$ prior gives $\alpha = 1$, $\beta = n + 1$. For $g = 1$, before adding the denominator stabilizer or batch normalization, Eq. (5) therefore gives

$$a_0^{\mathrm{post}} = -\sqrt{\frac{n+3}{n+1}}, \qquad a_1^{\mathrm{post}} = +\sqrt{\frac{n+3}{n+1}} \quad \text{for an all-correct history.} \tag{19}$$

Thus $|a^{\text{post}}| \leq \sqrt{2}$ in these two cases, and adding $\delta > 0$ can only reduce the magnitude. This follows by substituting the posterior mean and standard deviation and cancelling the factor $n + 2$. Without standardization, the corresponding scores are $-1/(n + 2)$ and $+1/(n + 2)$.

The condition concerns *all generated observations*, not just the retained pool. An all-correct rescue round can follow an all-incorrect probe, so its posterior history is mixed and this bound does not apply. Nor does a finite score bound the policy gradient: token score vectors, importance ratios, and clipping still matter. Posterior counts are a statistical model for a fixed query and behavior policy; unrelated queries must not share counts through reused batch-position identifiers.

C.5 Mean and Variance of Uniform Negative Subsampling

Fix a sampled pool with $c > 0$ correct responses and $f > 1$ incorrect responses, and fix their vectors v_i from Eq. (6). Select m of the f incorrect responses uniformly without replacement, retaining all c correct responses. Use the same inactive-clipping, zero-KL, no-batch-normalization, population-standard-deviation assumptions as Theorem C.1. Let $\bar{v}_I$ be the mean over all f incorrect vectors and define

$$a_m = \frac{\sqrt{cm}}{c+m}, \qquad \Sigma_I = \frac{1}{f-1}\sum_{i \in I}(v_i - \bar{v}_I)(v_i - \bar{v}_I)^\top.$$

Then, with expectation only over subset selection,

$$\mathbb{E}_S[\widehat{G}_m] = a_m(\bar{v}_C - \bar{v}_I), \tag{20}$$

$$\mathrm{Cov}_S(\widehat{G}_m) = a_m^2 \left(\frac{1}{m} - \frac{1}{f}\right)\Sigma_I. \tag{21}$$

Proof. Each incorrect response is included with probability m/f, so the selected mean has expectation $\bar{v}_I$. For distinct indices, joint inclusion has probability $m(m-1)/(f(f-1))$. Expanding the second moment of the selected mean with these two inclusion probabilities gives $(1/m - 1/f)\Sigma_I$. Multiplication by a_m in Eq. (6) gives the two claims. □

The full-pool gradient is $a_f(\bar{v}_C - \bar{v}_I)$, so the expected subsampled gradient is a positive rescaling of it for this fixed pool. Its covariance is generally nonzero when $m < f$, unlike retaining the whole fixed pool. Consequently,

$$\mathbb{E}_S\|\widehat{G}_m\|^2 = a_m^2 \left(\|\bar{v}_C - \bar{v}_I\|^2 + \left(\frac{1}{m} - \frac{1}{f}\right)\mathrm{tr}\,\Sigma_I\right).$$

A larger second moment can therefore reflect added noise. The rescaling varies across query pools and is not an unbiased recovery of the original training objective. These expressions do not cover length-based selection, active clipping, posterior-scored homogeneous groups, or stochasticity from generating a different pool.

C.6 Choosing Allocation Parameters

The probe size and rescue cap trade additional generation against the chance of observing a success. Under conditionally independent $\mathrm{Bernoulli}(p)$ outcomes for a fixed policy, the all-incorrect probe probability is $(1-p)^{n_{\text{explore}}}$. After an all-incorrect probe, a fresh attempts find at least one success with probability $1-(1-p)^a$. This calculation explains why small increments limit work on queries rescued early, while larger increments can reduce sequential launch overhead. It does not account for variable response lengths or a shared batch cap.

We use half of the nominal response allowance for the probe, $S = 0$ to resample all-incorrect groups, and $k = 1$ as the smallest balanced mixed subset. The hard caps B, $n_{\max}$, and $K_{\max}$ bound generation even for queries that never succeed; h bounds the number of homogeneous responses used for an update. These are practical operating choices. The existing k sweep measures update cost, not a comprehensive sensitivity analysis of the probe, rescue, prior, and retention parameters.

C.7 Multi-Turn Plot and Throughput Accounting

Table 1E reports the five $b = 16$ run summaries. The throughput numerator is the last logged rollout-step counter, not the number of trajectories or optimizer minibatches. The highlighted comparison targets four mixed groups for adaptive allocation and eight for the GRPO $N = 16$ reference. Values retain the source report's rounding. Run durations and evaluation counts differ, so peak scores are not equal-budget final scores or independent-seed estimates.

Figure 2 shows the same five runs, with all 131 recorded evaluations retained. Its time axis starts at each run's first evaluation, not at scheduler launch. Faint lines connect the observations; bold lines apply a centered three-point mean, using the available points at each endpoint. No running maximum, confidence band, additional seed, or interpolated evaluation is introduced.

D Supplementary Experiments

D.1 Datasets

The datasets below cover mathematical reasoning and code generation.

Table 3: Overview of Training and Evaluation Datasets

Dataset	Type	Description
Training Dataset (Math Reasoning)		
OpenR1-AERO	Training	40k math problems curated from OpenR1-Math-220k (Open R1, 2025).
Evaluation Datasets		
AIME (2024)	Evaluation	American Invitational Mathematics Examination; competition-level high school math.
AMC (2023)	Evaluation	American Mathematics Competitions; qualifier benchmark for AIME.
MATH-500	Evaluation	500-problem subset of the MATH benchmark, spanning pre-algebra to calculus (Lightman et al., 2023).
MinervaMath	Evaluation	Quantitative reasoning problems across STEM fields (Lewkowycz et al., 2022).
OlympiadBench	Evaluation	National and international Olympiad problems requiring advanced reasoning (He et al., 2024).
Training Dataset (Code Generation)		
Prime-1.4K	Training	1.4k curated code data from the original Eurus-45k, used by Cui et al. (2025).
Evaluation Datasets		
APPS	Evaluation	Hendrycks et al. (2021)
CodeContests	Evaluation	Li et al. (2022)
TACO	Evaluation	Li et al. (2023).

D.2 Baselines and Benchmarks

D.2.1 Evaluation Metrics

All policies are evaluated on the same held-out problem set $\mathcal{E}$, with n fresh responses per problem. The evaluation sample count is independent of training-time group size or filtering.

Primary Performance Metrics:

- **Pass@n:** Fraction of held-out problems with at least one correct response among the n evaluation samples:

$$\text{Pass@}n = \frac{1}{|\mathcal{E}|} \sum_{p \in \mathcal{E}} \mathbb{I}\left[\max_{1 \le i \le n} c_{p,i} = 1\right]. \tag{22}$$

  Here $c_{p,i} \in \{0, 1\}$ denotes the correctness of evaluation response i for problem p.

- **Avg@n:** Mean correctness across the same held-out problems and fresh responses:

$$\text{Avg@}n = \frac{1}{|\mathcal{E}|} \sum_{p \in \mathcal{E}} \frac{1}{n} \sum_{i=1}^{n} c_{p,i}. \tag{23}$$

  This estimates single-response correctness of the trained policy, not the performance of the training-time allocation rule.

- **Computational Efficiency:** Total FLOPs per step including rollout generation and training:

$$\text{Total FLOPs} = \text{FLOPs}_{\text{rollout}} + \text{FLOPs}_{\text{training}} \tag{24}$$

  The parameter–token approximation uses separate token counts for generation and model updates:

$$F_{\text{gen}} \approx 2N_{\text{params}}T_{\text{gen}}, \qquad F_{\text{update}} \approx 6N_{\text{params}}T_{\text{update}}. \tag{25}$$

Here N_{params} is the parameter count. T_{gen} counts tokens processed for generation, including responses later discarded, while T_{update} counts tokens processed by model updates on retained responses. These are FLOP estimates rather than hardware measurements. Equal nominal rollout caps do not imply equal realized token counts.

D.3 Additional Results

D.3.1 Additional Budget Baseline Results

Table 4 presents additional budget baseline comparisons for Qwen2.5-7B-Math and Qwen2.5-1.5B-Base models. Consistent with the main results, AERO achieves the best efficiency-performance trade-off across all model configurations. For the 7B model, AERO matches the training time of the reduced-budget baseline ($N = 6$) while achieving significantly higher accuracy (Avg@8: 0.48 vs. 0.455). The advantage is even more pronounced on the Base model, where AERO delivers a 4× improvement in Avg@8 compared to the time-matched $N = 5$ baseline (0.234 vs. 0.056), demonstrating that adaptive rollout allocation is particularly beneficial for weaker models that produce more zero-accuracy queries.

D.3.2 Downsampling Ratio Cost Analysis

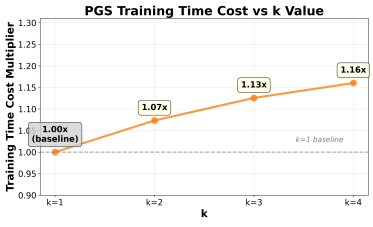


Figure 9: Training Time Cost vs. k

Table 4: Computational budget and alternative sampling strategies. The best-performing method is highlighted in blue, and time-matched baselines for comparison are in light blue.

Method	**Time (min)**	**FLOPs (Total)**	**Avg@8** (P-value)	**Pass@8**
Qwen2.5-1.5B-Math				
GRPO	34.3	3627	0.3401 (= 0.28)	0.6090
DAPO	26.1	2722	0.3405 (= 0.31)	0.6100
AERO	**19.1**	**1811**	**0.3452**	**0.6121**
N/2	23.7	1863	0.3211 (= 0.004)	0.5954
0.32N	20.1	1133	0.3184 (= 0.002)	0.5891
N/4	16.3	906	0.3133 (= 0.001)	0.5870
N/8	8.9	453	0.2820 (< 1e-4)	0.5791
Base	—	—	0.1617 (< 1e-6)	0.5050
Qwen2.5-7B-Math				
AERO	10.8	9181	0.4800	0.7000
DAPO	15.8	13743	0.4750 (= 0.29)	0.6910
GRPO	21.5	17334	0.4690 (= 0.05)	0.6860
N=6	10.6	6500	0.4550 (= 0.003)	0.6790
Base	—	—	0.3110(< 1e-6)	0.6220
Qwen2.5-1.5B-Base				
AERO	19	1849	0.2340	0.4990
DAPO	23.8	2289	0.2260 (= 0.12)	0.4770
GRPO	36.4	3714	0.2120 (= 0.006)	0.4460
N=5	19.8	1160	0.0560 (< 1e-6)	0.2510
Base	—	—	0.0340(< 1e-6)	0.1820

Figure 9 shows that the training-time cost rises steadily as the downsampling ratio k increases: a 7% increase from $k = 1$ to $k = 2$, 13% from $k = 1$ to $k = 3$, and 16% from $k = 1$ to $k = 4$.

D.3.3 Bayesian Stabilization: Full Training Dynamics

Table 5 and Figure 10 report gradient norms during training. The Bayesian-score variant has approximately $1.5\times$ larger recorded norms; the table reports $p < 0.001$ for the norm comparison. These measurements describe magnitude, not gradient variance, alignment with an unfiltered update, or a guarantee of stable convergence.

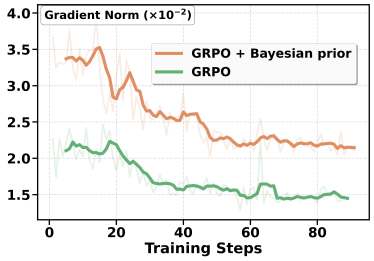


Figure 10: Gradient norm: Bayesian vs GRPO.

Table 5: Actor gradient norm ($\times 10^{-2}$) at steps 70/80/90 for GRPO + Bayesian Prior and GRPO. The Bayesian variant shows $\sim 1.5\times$ larger gradient norms.

	Grad Norm ($\times 10^{-2}$)			
Method	70	80	90	p**-value**
GRPO + Bayesian Prior	2.21	2.22	2.10	—
GRPO	1.46	1.38	1.40	< 0.001

D.3.4 Effect of Discarding High-Success Rollouts

We conduct rejection sampling on high-success partial problems ($u \geq 0.5$). For example, consider a query with 8 rollouts, 5 correct and 3 incorrect. Under a $1:1$ scheme, we retain 3 correct and 3 incorrect rollouts. Compared to AERO (Table 6), training time is nearly identical (19.0 vs. 19.1 minutes per step) with similar estimated FLOPs (1805 vs. 1811 PFLOPs), not rollout counts. However, accuracy drops noticeably, with Avg@8 falling from 0.3452 to 0.3321 and Pass@8 from 0.6121 to 0.6100 (Table 6). This shows that discarding correct rollouts from high-success queries degrades performance without yielding significant cost savings.

Table 6: Efficiency and performance with rejection sampling on queries with $u \geq 0.5$.

Method	Time (min)	FLOPs (PetaFLOP)	Avg@8	Pass@8
Qwen2.5-1.5B-Math				
GRPO	34.3	3627	0.3401	0.6090
DAPO	26.1	2722	0.3405	0.6100
AERO	19.1	1811	**0.3452**	**0.6121**
AERO (rejection sampling, $u \geq 0.5$)	19.0	1805	0.3321	0.6100

E Learning Framework

We implement AERO using the verl framework (Sheng et al., 2025), a distributed RL system designed for LLMs. Our approach employs GRPO with specialized resource allocation mechanisms.

Distributed Architecture: The framework utilizes a Ray-based single controller design with multiple worker groups:

- **ActorRollout Worker:** A hybrid worker that combines policy optimization (actor) and sequence generation (rollout) using FSDP (Fully Sharded Data Parallel) for efficient memory usage.
- **Critic:** No critic or learned value network is used in these GRPO runs.
- **Reference Policy Worker:** Maintains the reference policy π_{ref} for KL regularization (optional).

Model Architecture (Math & Code): We train and evaluate AERO on multiple Qwen2.5-family backbones:

- **Math models:** Qwen2.5-Math-1.5B and Qwen2.5-Math-7B.
- **Code model:** Qwen2.5-7B-Instruct-1M (code generalization setting).

All models are trained with:

- **Precision:** bfloat16 mixed precision.
- **Parallelization:** FSDP sharding (configured per model scale).
- **Memory optimization:** gradient checkpointing and padding removal.

The policy outputs token probabilities:

$$\pi_\theta(a_t \mid s_t) = \mathrm{softmax}(f_\theta(s_t)) \,. \tag{26}$$

AERO: AERO implements adaptive resource allocation through iterative rejection sampling:

1. **Problem Categorization:** Each problem p is classified based on current accuracy $\bar{c}_p$ with rebalance threshold $\tau = 0.5$:

$$\text{Category}(p) = \begin{cases} \text{Zero} & \text{if } \bar{c}_p = 0, \\ \text{Partial} & \text{if } 0 < \bar{c}_p < \tau, \\ \text{Good} & \text{if } \tau \leq \bar{c}_p < 1, \\ \text{Perfect} & \text{if } \bar{c}_p = 1. \end{cases} \tag{27}$$

2. **Rescue and Retention:** Generation and retention have separate counters. In the default $S = 0$ math setting, unresolved all-incorrect queries receive at most $\Delta n = 2$ additional responses per round, for at most $K_{\max} = 10$ rounds:

$$N_{p,\text{gen}}^{(j+1)} = \begin{cases} N_{p,\text{gen}}^{(j)} + \Delta n_p^{(j)}, & \text{if } p \text{ remains eligible for rescue,} \\ N_{p,\text{gen}}^{(j)}, & \text{otherwise,} \end{cases} \tag{28}$$

where $0 \le \Delta n_p^{(j)} \le \Delta n$ also respects the remaining per-query and global allowances. The retained count $N_{p,\text{train}}$ is computed after subset selection and does not subtract from the generation counter.

3. **Budget Constraints:** Sample allocation respects both per-problem and total limits:

$$N_{p,\text{gen}} \le n_{\max} = 32 \quad \text{for every query } p, \tag{29}$$

$$\sum_{p=1}^{P} N_{p,\text{gen}} \le B = n_{\text{total}} P, \qquad n_{\text{total}} = 16. \tag{30}$$

These bounds count all generated responses, including discarded ones.

Pooling and Selection. A rescued query supplies its successful rescue round as the candidate pool. With `rescued_zero_combine_original` enabled, that round is combined with the original probe; failed intermediate rounds still count toward generation and posterior statistics. An unresolved all-incorrect query supplies its probe pool for homogeneous retention. The `prefer_shortest` and `prefer_shortest_incorrect` switches choose an ordered subset; the corresponding disabled branches sample uniformly. In the inspected source, the ordering key is `responses[i].size(-1)`, the tensor width rather than the unpadded response length. Equal-width padded responses therefore retain their original order under the stable sort. This is not an informativeness ranking, and the uniform-subset analysis does not apply to this ordered branch.

E.1 Normalization and Run Correspondence

Group standardization and batch normalization are separate switches. First form preliminary scores $a_{q,i}$ using posterior statistics for homogeneous retained groups and empirical statistics for mixed groups. The flag `grpo_use_std` (g) controls division by the corresponding group standard deviation plus δ. The flag `advantage_batch_std` (w) then gives

$$\hat{A}_{q,i} = \begin{cases} (a_{q,i} - \bar{a})/(s_a + \delta), & w = 1, \\ a_{q,i}, & w = 0, \end{cases} \qquad \bar{a} = \frac{1}{M}\sum_{j=1}^{M} a_j, \quad s_a^2 = \frac{1}{M-1}\sum_{j=1}^{M}(a_j - \bar{a})^2. \tag{31}$$

Here $M > 1$ is the update-batch size. The archived implementation computes these statistics after padding the batch to the distributed worker count, so padding entries participate in this normalization. Group empirical standard deviations also use the sample convention. The population standard deviation in Theorem C.1 is an explicit analytical assumption, not a description of this batch operation.

The saved evaluation entry for the 1.5B AERO row has Avg@8 0.345151 and Pass@8 0.612096 at step 90, matching the paper's rounded $0.3452/0.6121$ and its five benchmark columns. Its adjacent launch-script record sets $g = 0, w = 1$, `rescued_zero_combine_original=False`, and both ordered-selection flags to `True`. It uses $k = 1$, probe size 8, rescue increment 2, global allowance $16P$, $n_{\max} = 32$, ten rescue rounds, and four retained responses for homogeneous groups. Thus the posterior preliminary score in this configuration is $r - \tilde{u}$, followed by batch normalization; it is not the posterior-standardized $g = 1$ variant. Local run archives corroborate these switches through step 80, but do not contain the matching step-90 held-out evaluation. The same run identifier also appears in later launch records with different ablation switches and no recorded evaluations. Run identifiers alone therefore do not identify a unique configuration. These records request four GPUs, so they do not independently verify the separately reported two-A10 timing setup. A checkpoint-to-configuration mapping remains necessary to reproduce the full results table.

Bayesian Posterior Updates (Robust Success-Rate Estimation). AERO maintains a Bayesian posterior over each prompt's success probability u to obtain a robust estimate even in all-fail ($c = 0$) or all-correct ($c = n$) cases. We place a Beta prior on u,

$$u \sim \mathrm{Beta}(\alpha_0, \beta_0), \tag{32}$$

and observe c correct rollouts out of n samples. By Beta–Binomial conjugacy, the posterior is

$$p(u \mid c, n) \sim \mathrm{Beta}(\alpha_0 + c,\ \beta_0 + n - c). \tag{33}$$

We use the posterior mean as a smoothed success-rate estimate:

$$\tilde{u} \ = \ \mathbb{E}[u \mid c, n] \ = \ \frac{c + \alpha_0}{n + \alpha_0 + \beta_0}, \tag{34}$$

where we set a weak prior $\alpha_0 = \beta_0 = 1$ in practice. This posterior smoothing ensures $\tilde{u} \in (0, 1)$ even when $c \in \{0, n\}$. For homogeneous retained groups, the standardized score also uses the posterior standard deviation in Eq. (4), as specified by Eq. (5); replacing the mean alone would not resolve a zero empirical denominator. In this paragraph, c and n count generated outcomes, not only retained responses.

GRPO Training: The framework uses GRPO without critic networks:

$$\mathcal{L}_{\text{GRPO}} = \mathbb{E}_{(s,a)\sim\pi_{\theta_{\text{old}}}} \left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right] \tag{35}$$

where advantages A_t are computed using group-relative baselines rather than learned value functions, and $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance ratio.

Hyperparameter Configuration. Table 7 preserves the reported math and code settings. The separately identified normalization and selection switches are described in Appendix E.1; a single "advantage std" flag does not specify both normalization operations.

F Data Collection and Prompt Engineering

AERO operates on structured mathematical reasoning datasets with systematic prompt engineering and automatic reward computation. Training data is collected on-policy during the GRPO training loop, with careful control over prompt design and reward evaluation.

System Prompt Design.

Math System Prompt: "Your task is to solve the given question step by step. You should conduct a systematic, thorough reasoning process before providing the final answer. This involves analyzing, summarizing, exploring, reassessing, and refining your reasoning process through multiple iterations. Each reasoning step should include detailed analysis, verification, and refinement of ideas. Include the final answer in `\boxed{}` for closed-form results such as multiple-choice or numerical answers."

Code System Prompt: "When tackling complex reasoning tasks, you have access to the following actions. Use them as needed to progress through your thought process.
[ASSESS] [ADVANCE] [VERIFY] [SIMPLIFY] [SYNTHESIZE] [PIVOT] [OUTPUT]
You should strictly follow the format below:
[ACTION NAME] Your action step 1 Your action step 2 ...
Next action: [NEXT ACTION NAME]
After reasoning, output the final executable Python code enclosed in a single code block, e.g.,
```python
*your solution code*
```"

Table 7: AERO hyperparameters for math and code experiments.

Parameter	Math	Code
	Model / System	
Backbone	Qwen2.5-Math-(1.5B/7B)	Qwen2.5-7B-Instruct-1M
Precision	bfloat16	bfloat16
FSDP Sharding	Configured per run	Configured per run
Grad Checkpointing	True	True
	Optimization	
Learning rate	5e-6	5e-6
Warmup ratio	0.03	0.03
Grad clipping	1.0	1.0
Clip ratio ϵ	0.2	0.2
Entropy coef.	1e-3	1e-3
GRPO epochs	1	1
	AERO Sampling	
n_{explore}	8	8
n_{total}	16	16
$n_{\max}$	32	32
Δn	2	4
$K_{\max}$	10	3
Threshold τ	0.5	0.5
	Sequence / Data	
Train batch size	256	128
Val batch size	150	200
Max prompt length	1024	1024
Max response length	1536	1536
Sampling temperature	1.0	1.0
	Bayesian Smoothing	
α_0, β_0	1, 1	1, 1

Example Problem. A representative training sample is shown below:

User Query: "A ship traveling along a river has covered 24 km upstream and 28 km downstream. For this journey, it took half an hour less than for traveling 30 km upstream and 21 km downstream... Determine the speed of the ship in still water and the speed of the river."
Expected Response: "Let v be the speed of the ship in still water... [step-by-step reasoning] ... The speed of the ship in still water is $\boxed{10}$ km/h and the speed of the river is $\boxed{4}$ km/h."

Example Problem.
A representative training sample (code) is shown below:

User Query: "Given base n and a string k consisting of digits in $[0, n-1]$, interpret k as the concatenation of base-n digits written in decimal (digits ≥ 10 are written as their decimal numerals without separators). Find the minimum $x \leq 10^{18}$ whose base-n representation produces exactly k."
Expected Response:

```
[ASSESS]
Parse k into tokens (each token is one base-n digit written in decimal,
no leading zeros unless token=="0").
Next action: [ADVANCE]

[ADVANCE]
DP over positions: dp[i]=min value for prefix k[0:i].
Transition by choosing next token k[i:j] as digit d<n:
dp[j]=min(dp[j], dp[i]*n + d), discard if >1e18.
Next action: [VERIFY]

[VERIFY]
Enforce token rules (no leading zeros), and cap transitions at 1e18.
Next action: [OUTPUT]

[OUTPUT]
```python
# ... final accepted solution ...
```

On-Policy Data Collection. During training, AERO collects experience tuples $(s_t, a_t, r_t, \log \pi_\theta(a_t|s_t))$:

- s_t: Formatted prompt containing the system instruction and question
- a_t: Generated reasoning trace following the structured format
- r_t:$r_t \in \{0, 1\}$, the Binary correctness reward
- $\log \pi_\theta(a_t|s_t)$: Log-probability of the generated trajectory

G Reward System

G.1 Multi-Domain Reward Framework

AERO uses verifiable, domain-specific evaluators to provide reliable rewards across heterogeneous reasoning tasks. Given a model response s and ground-truth answer g, the reward is computed by invoking the appropriate verifier for the task domain:

$$r_t = \mathcal{V}_d(s, g), \qquad d \in \{\text{math}, \text{code}, \text{science/QA}\}, \tag{36}$$

where $\mathcal{V}_{\text{math}}$ checks mathematical equivalence, $\mathcal{V}_{\text{code}}$ executes unit tests for program synthesis, and $\mathcal{V}_{\text{science/QA}}$ performs answer matching for structured QA (e.g., multiple-choice). Unless otherwise specified, we use a binary reward $r_t \in \{0, 1\}$ indicating whether the response is verified as correct.

G.2 Token-Level Reward Assignment

Following standard RL practice, rewards are assigned sparsely to the final token of each response:

$$\text{reward}[i, \text{len} - 1] = r_t. \tag{37}$$
```