Efficient Gradient-Aware Asynchronous Reinforcement Learning for LLM Post-Training

Songhan Yang[1], **Jie Wang**[1,*], **Yinqi Bai**[1], **Xialiang Tong**[2]
Jianye Hao[3], **Mingxuan Yuan**[2], **Feng Wu**[1]

[1]MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China, China
[2]Huawei Technologies Co., Ltd., China
[3]College of Intelligence and Computing, Tianjin University, China

[*]Corresponding author: jiewangx@ustc.edu.cn

Abstract

Asynchronous reinforcement learning (RL) improves the efficiency of large language model (LLM) post-training by separating rollout generation from policy optimization. However, this process produces stale off-policy samples, making effective replay selection important for learning. Replay selection methods such as D-ARL evaluate candidates under the current policy, which increases computational cost. We propose **G**radient-**A**ware **A**synchronous **R**einforcement **L**earning (GA-ARL), a framework for efficient replay selection. GA-ARL derives an analytic target distribution from the KL-regularized RL objective and weights it by absolute advantage. This design combines the policy preference captured by the analytic distribution with the learning signal reflected in advantage magnitude. GA-ARL computes replay scores using cached behavior-policy and reference-policy probabilities and advantages. It selects samples with low discrepancy between their behavior probabilities and the target scores, without additional current-policy evaluation during selection. Across five reasoning benchmarks, GA-ARL improves average accuracy over D-ARL by 3.1% on Qwen3-1.7B and 1.4% on Qwen3-4B in relative terms. On Qwen3-1.7B, it also reduces wall-clock training time by 25.3% on average.

1 Introduction

Reinforcement learning (RL) has become a widely used post-training approach for large language models (LLMs), enabling models to improve reasoning, alignment, and instruction-following performance through reward-guided optimization (Ouyang et al., 2022; Shao et al., 2024). Most existing RL methods for LLM post-training follow a synchronous training pipeline, where rollout generation and policy optimization are performed alternately (Schulman et al., 2017; Konda & Tsitsiklis, 1999). Although this paradigm ensures that each update uses trajectories sampled from the latest policy, it prevents data collection and model optimization from being fully overlapped. As a result, synchronous RL often suffers from low resource utilization and limited training efficiency, especially when both rollout generation and policy updating are computationally expensive (Uehara et al., 2022).

Asynchronous RL addresses this limitation by decoupling data generation from policy optimization (Lu et al., 2025; Noukhovitch et al., 2025b). By allowing rollout workers and training workers to execute in parallel, asynchronous pipelines reduce idle time and improve hardware utilization (Noukhovitch et al., 2025a; Hilton et al., 2022; Chen et al., 2025). However, this decoupled design introduces stale off-policy samples, since replay data may be generated by earlier behavior policies rather than the current policy. Such distribution mismatch can increase gradient variance, destabilize policy updates, and degrade the reasoning performance of LLMs.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To mitigate this issue, D-ARL (Bai et al., 2026) , a state-of-the-art asynchronous RL method, maintains a replay buffer from recent behavior policies and selects samples expected to reduce optimization variance. However, it requires evaluating candidate replay samples under the current model, causing substantial overhead for long responses and large buffers. Moreover, existing asynchronous RL methods mainly rely on variance reduction or current-policy matching for replay selection. Since the current policy is still evolving, it may not be the most desirable target for the next update, and samples selected only by these criteria may not provide the most effective training signals.

Motivated by these observations, we propose **Gradient-Aware Asynchronous Reinforcement Learning (GA-ARL)**, a lightweight replay selection framework for asynchronous RL. GA-ARL derives an analytic target distribution from the KL-regularized RL objective, capturing the policy preference induced by the advantage signal and reference-policy constraint. Reweighting this distribution by $|A(s,a)|$ incorporates the strength of gradient-based learning signals. In practice, GA-ARL scores replay candidates using cached behavior-policy and reference-policy probabilities, advantages, and a historical scaling estimate. It aggregates token-level discrepancies within responses and prompt groups, then selects low-discrepancy groups for policy optimization without additional current-policy evaluation during replay scoring. GA-ARL modifies only replay selection and integrates into asynchronous RL pipelines without changing the underlying RL objective.

Experiments on five mathematical, logical, and code reasoning benchmarks with Qwen3-1.7B and Qwen3-4B (Team, 2025) show that GA-ARL achieves the best average accuracy among the evaluated asynchronous baselines. Relative average-accuracy gains over ARLHF are 6.1% and 5.1% on the two model sizes, and gains over D-ARL are 3.1% and 1.4%, respectively. On Qwen3-1.7B, GA-ARL reduces wall-clock training time by 25.3% on average compared with D-ARL. Eight-seed evaluations on GSM8K, LiveCodeBench, and HumanEval show statistically significant mean accuracy gains over D-ARL alongside shorter training times. Additional comparisons cover synchronous GRPO and reward noise. With a maximum response length of $12{,}288$ tokens on Qwen3-4B LiveCodeBench, GA-ARL reaches 62.65% accuracy versus 58.38% for ARLHF under the same 66-hour wall-clock budget.

We summarize our main contributions as follows: (1) We identify two key challenges in asynchronous RL for LLM post-training: distribution shift caused by stale off-policy replay samples, and the high cost of current-model evaluation during replay selection. (2) We derive an analytic target distribution from the KL-regularized RL objective and introduce an advantage-weighted variant that accounts for both policy preference and gradient strength. (3) We propose GA-ARL, a lightweight replay selection framework that selects samples according to their consistency with the advantage-weighted analytic target, without modifying the underlying RL objective. (4) Experiments across five reasoning benchmarks and two model sizes demonstrate improved accuracy and wall-clock efficiency over asynchronous baselines, supported by eight-seed evaluations, a synchronous GRPO comparison, reward-noise tests, and long-response evaluation.

Code is publicly available at `https://github.com/MIRALab-USTC/GA-ARL`.

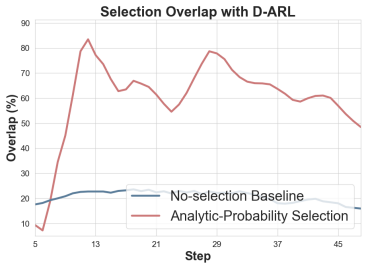
2 Background and Related Work

Notation. We denote a prompt by x and a response by $\tau = (a_1, \ldots, a_N)$, where a_i is the i-th generated token and s_i is its prefix state. For a policy π, $\pi(\tau|x) = \prod_{i=1}^{N} \pi(a_i|s_i)$. Let π_{θ_k} denote the policy at training round k. Replay samples in asynchronous RL may come from previous behavior policies, e.g., $\pi_{\theta_{k-1}}, \ldots, \pi_{\theta_{k-4}}$. We use π_θ for the current policy before a local update, π_ϕ for the policy being optimized, and π_{ref} for the reference policy. The replay buffer is $\mathcal{B}$ and $A(s,a)$ denotes the advantage.

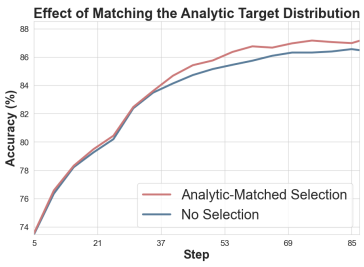
2.1 KL-regularized RL Objective

In LLM post-training, policy updates are commonly regularized by a reference policy π_{ref} to stabilize reward-driven optimization. Given samples from the current policy π_θ, the local KL-regularized surrogate loss is

$$L(\pi_\phi) = -\mathbb{E}_{s,a\sim\pi_\theta}\left[\frac{\pi_\phi(a|s)}{\pi_\theta(a|s)}A(s,a)\right] + \beta\mathbb{E}_{s\sim\pi_\theta}\left[KL\left(\pi_\phi(\cdot|s)\|\pi_{\mathrm{ref}}(\cdot|s)\right)\right]. \tag{1}$$



(a) Selection overlap with D-ARL.



(b) Effect of analytic target matching.

Figure 1: Motivating results for GA-ARL. (a) Analytic-probability-based selection largely recovers D-ARL-like sample preferences without current-policy evaluation. (b) Analytic-target-matched samples achieve better training performance than unfiltered replay samples.

The coefficient $\beta > 0$ controls the strength of KL regularization. For each state s, the corresponding KL-regularized optimization admits the closed-form optimal policy (Rafailov et al., 2023):

$$\pi^*(a|s) = \frac{\pi_{\text{ref}}(a|s)\exp(A(s,a)/\beta)}{Z(s)}, \quad Z(s) = \sum_{a'} \pi_{\text{ref}}(a'|s)\exp(A(s,a')/\beta). \tag{2}$$

Thus, the desired update direction is jointly shaped by the advantage signal and the reference-policy constraint.

2.2 Asynchronous Reinforcement Learning

Most RL methods for LLM post-training use synchronous pipelines, where rollout generation and policy optimization proceed sequentially (Zhang et al., 2025). This ensures fresh on-policy data but limits throughput due to tight coupling between data collection and optimization. Asynchronous RL methods, such as ARLHF (Noukhovitch et al., 2025a), Decoupled (Hilton et al., 2022), and CISPO (Chen et al., 2025), improve efficiency by decoupling these stages: rollout workers generate samples using behavior policies from different rounds, while training workers update the current policy in parallel. However, this design introduces stale off-policy data, since training samples may be generated by earlier policies rather than the current one. Existing methods typically use importance sampling (Hanna et al., 2021) to correct this mismatch; details are provided in Appendix C.1.

2.3 Distribution Shift and Replay Sample Selection in Asynchronous RL

Freshness-based replay selection. Asynchronous RL inevitably introduces distribution shift between replay data and the current training policy (Meng et al., 2023). A common strategy is to control data freshness by maintaining a bounded replay buffer and using samples generated by recent behavior policies (Fu et al., 2025; Lu et al., 2025). While this improves stability and throughput, freshness-based selection does not explicitly measure distribution mismatch and may discard useful samples from earlier policies.

Off-policy correction under stale behavior data. Another line of work focuses on off-policy correction under stale behavior data. Since replay samples may be generated by earlier behavior policies, direct importance sampling can suffer from high variance when the behavior policy deviates from the current policy. Existing methods mainly fall into two categories. Gradient truncation methods, such as Decoupled PPO (Hilton et al., 2022), stabilize training by truncating or clipping gradients from highly off-policy samples. Importance sampling optimization methods, such as CISPO (Chen et al., 2025), TOPR (Roux et al., 2025), and Truncated IS (Espeholt et al., 2018), reduce IS variance by truncating, reweighting, or normalizing importance ratios. While these methods improve off-policy training stability, they mainly correct or constrain samples during optimization, rather than explicitly selecting replay samples most useful for the next update. Thus, replay selection under distribution mismatch remains insufficiently addressed.

Distribution-aware replay selection. D-ARL (Bai et al., 2026), a state-of-the-art replay selection method, addresses this issue by maintaining samples from multiple recent behavior policies and

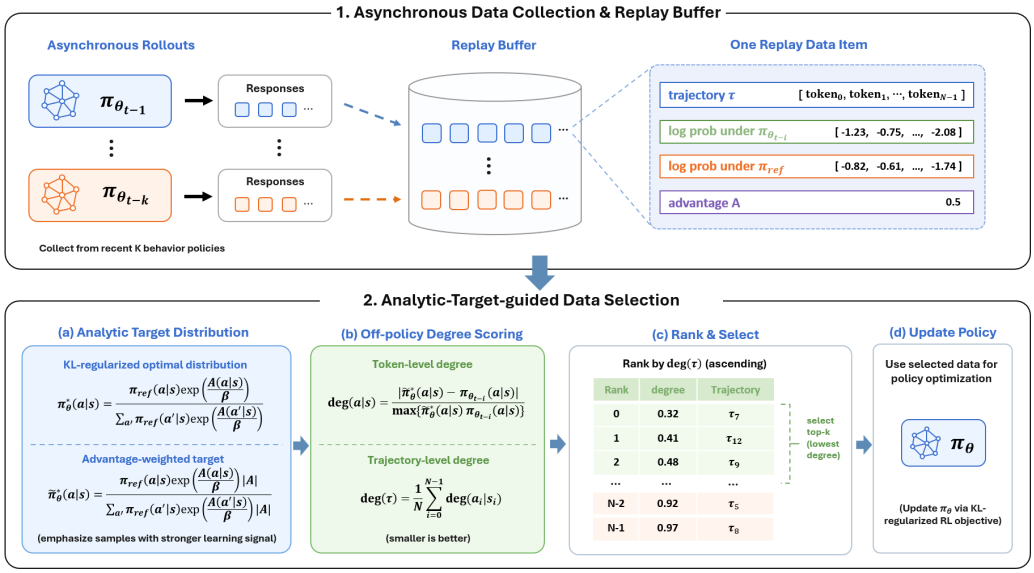


Figure 2: Overview of GA-ARL. GA-ARL stores replay samples from recent behavior policies with rollout-policy and reference-policy probabilities, computes their discrepancy to an advantage-weighted analytic target, and selects low-discrepancy samples for policy optimization.

selecting samples expected to reduce the variance of policy optimization. However, D-ARL requires evaluating candidate replay samples under the current policy, which introduces substantial overhead for long responses and large replay buffers. Moreover, variance reduction or current-policy matching may not fully indicate whether a sample aligns with the objective-preferred update direction. These limitations motivate more efficient and objective-aware replay sample selection for asynchronous RL. More details of D-ARL are provided in Appendix C.2.

3 Motivating Results

3.1 Analytic Probability Recovers D-ARL-like Sample Preference

D-ARL selects replay samples using a variance-aware metric that requires evaluating candidate responses under the current policy, which introduces additional inference overhead in long-response LLM training. Motivated by the analytic solution of the KL-regularized RL objective, we replace the current-policy probability in the D-ARL metric with the analytic probability and examine whether it induces a similar sample preference. As shown in Figure 1(a), analytic-probability-based selection achieves around 60% overlap with D-ARL, substantially higher than the no-selection baseline of about 25%. This indicates that analytic probability can preserve a large portion of D-ARL's variance-aware selection behavior while avoiding additional current-policy evaluation.

3.2 Analytic Target Matching Improves Training

The analytic solution induced by the KL-regularized objective characterizes the policy preference of the current update. To examine whether this distribution can serve as a useful replay selection target, we compare training with unfiltered replay samples and training with samples that have smaller discrepancy to the analytic target distribution. As shown in Figure 1(b), analytic-matched selection achieves consistently better performance than no selection across training steps. This suggests that replay samples closer to the analytic target provide more effective training signals, motivating the use of analytic-distribution-guided replay selection in GA-ARL.

4 Method

We propose **Gradient-Aware Asynchronous Reinforcement Learning (GA-ARL)**, a lightweight replay selection framework for asynchronous RL post-training. As shown in Figure 2, GA-ARL maintains replay samples from recent behavior policies and selects samples according to their discrepancy from an advantage-weighted analytic target distribution. Since GA-ARL only modifies the replay selection stage, it can be integrated into existing asynchronous RL pipelines without changing the underlying RL objective. More algorithm details are provided in Appendix F.4.

4.1 Replay Buffer Management

At training stage t, GA-ARL maintains a replay buffer containing samples from the most recent K behavior policies:

$$\mathcal{B}_t = \bigcup_{j=0}^{K-1} \mathcal{D}_{t-j}, \tag{3}$$

where $\mathcal{D}_{t-j}$ denotes rollout data generated by behavior policy $\pi_{\theta_{t-j}}$. Each replay response stores its prompt x, rollout-group identifier g, tokens τ, outcome reward R, behavior-policy version k, and token-level cached quantities:

$$(x, g, \tau, R, \{\ell_i^{\mathrm{b}}, \ell_i^{\mathrm{ref}}, m_i\}_{i=0}^{N-1}, k), \tag{4}$$

where $\ell_i^{\mathrm{b}} = \log \pi_{\theta_k}(a_i|s_i)$, $\ell_i^{\mathrm{ref}} = \log \pi_{\mathrm{ref}}(a_i|s_i)$, and m_i is the valid-response-token mask. Group identifiers preserve the responses generated together for each prompt; group advantages are computed from their stored rewards when the buffer is scored. GA-ARL computes reference-policy log-probabilities when each rollout block enters the buffer and caches them for reuse during replay scoring. Together with cached behavior-policy log-probabilities, these quantities enable replay scoring without additional current-policy evaluation. The buffer is updated in a FIFO manner at the policy-block level.

4.2 Advantage-weighted Analytic Target Distribution

D-ARL performs replay selection using a variance-aware metric, which requires the probability of each candidate sample under the current policy. This current-policy evaluation is expensive in LLM post-training, especially when responses are long or the replay buffer contains many candidates. Instead of scoring replay samples by repeatedly evaluating their probabilities under the current policy, GA-ARL constructs an objective-aware analytic target from the KL-regularized RL objective. Replay samples are then selected according to their discrepancy from this target, using cached reference-policy and behavior-policy probabilities. Based on this, GA-ARL uses the following advantage-weighted analytic target for replay selection:

$$\tilde{\pi}^*(a|s) = \frac{\pi_{\mathrm{ref}}(a|s)\exp(A(s,a)/\beta)|A(s,a)|}{\sum_{a'}\pi_{\mathrm{ref}}(a'|s)\exp(A(s,a')/\beta)|A(s,a')|}. \tag{5}$$

This analytic target motivates a replay scorer based on cached reference-policy probabilities and advantages, inducing sample preferences similar to D-ARL as shown in our motivating results. The following theoretical results provide further interpretation of why this is a reasonable and effective criterion for replay sample selection.

Theorem 4.1 (Analytic target policy induced by KL regularization)**.** *For a fixed state s, the KL-regularized state-level objective induces the following analytic target policy distribution:*

$$\pi^*(a|s) = \frac{\pi_{\mathrm{ref}}(a|s)\exp(A(s,a)/\beta)}{Z(s)}, \tag{6}$$

where

$$Z(s) = \sum_{a'} \pi_{\mathrm{ref}}(a'|s)\exp(A(s,a')/\beta). \tag{7}$$

This distribution characterizes the policy direction preferred by the current KL-regularized update.

Theorem 4.1 identifies the policy preference induced by the advantage and reference-policy regularization; its proof is given in Appendix B.1. Replay selection uses this preference to rank the samples contributing to a finite-batch update. Weighting by advantage magnitude further accounts for their gradient signal strength, as described below.

Proposition 4.2 (Advantage-weighted target for replay selection)**.** *Under a local update approximation and assuming comparable token-level score-function norms, the contribution of a replay sample* (s, a) *to the target-aligned policy update can be characterized by*

$$\pi^*(a|s)|A(s,a)|. \tag{8}$$

Under the local-update and comparable-score-norm assumptions, Proposition 4.2 motivates the target-side factor $\pi^*(a|s)|A(s,a)|$. The first term favors actions aligned with the KL-regularized policy preference, while the second increases the weight of stronger learning signals. Appendix B.2 relates this factor to the sample gradient and its behavior-policy importance ratio.

In GRPO training, response rewards are centered within each rollout group and divided by the group standard deviation plus 10^{-6}. The response advantage is broadcast to valid response tokens. For each sampled token, the implemented replay scorer uses

$$v_i = \ell_i^{\mathrm{ref}} + \frac{A_i}{\beta_{\mathrm{target}}} + \log|A_i + \epsilon| - \log|\widehat{z}_t + \epsilon|, \tag{9}$$

where $\beta_{\mathrm{target}} = 1$ is the scoring temperature, $\epsilon = 10^{-8}$, and $\widehat{z}_t$ is a shared scalar estimated from past training updates. The sampled-token target score is $\widehat{p}_i = \exp(v_i)$. Appendix F.2 gives the advantage and historical-scaling formulas.

4.3 Off-policy Degree with Respect to the Target Distribution

GA-ARL compares the sampled-token target score $\widehat{p}_i$ with the cached behavior probability. It computes the max-normalized difference directly in log space:

$$d_i = 1 - \exp\big(-|v_i - \ell_i^{\mathrm{b}}|\big) = \frac{|\widehat{p}_i - \pi_{\theta_k}(a_i|s_i)|}{\max(\widehat{p}_i, \pi_{\theta_k}(a_i|s_i))}. \tag{10}$$

The response score averages over valid tokens:

$$\deg(\tau) = \frac{\sum_i m_i d_i}{\max(1, \sum_i m_i)}. \tag{11}$$

A smaller score indicates closer agreement with the sampled-token target. This symmetric discrepancy lies in $[0, 1]$ and measures relative disagreement, preserving differences between small token scores that would be obscured by their absolute difference. Its log-space form uses the cached sampled-token quantities directly. Further discussion is provided in Appendix F.3.

4.4 Replay Selection and Policy Optimization

Let $\mathcal{G}_g$ contain the responses generated together for prompt group g. GA-ARL averages their response scores and selects M groups:

$$\deg(g) = \frac{1}{|\mathcal{G}_g|} \sum_{\tau \in \mathcal{G}_g} \deg(\tau), \qquad \mathcal{S}_t = \mathrm{TopM}_g(-\deg(g)). \tag{12}$$

The training batch contains all responses in these groups, $\mathcal{D}_{\mathrm{sel}} = \bigcup_{g \in \mathcal{S}_t} \mathcal{G}_g$, preserving the group structure used by GRPO. During warmup, $\mathcal{D}_{\mathrm{sel}}$ is the newly received rollout batch. Here K denotes the number of retained rollout blocks and M the number of selected prompt groups.
The actor is updated with the existing GRPO training objective:

$$\mathcal{L}_{\mathrm{actor}}(\phi; \mathcal{D}_{\mathrm{sel}}) = \mathcal{L}_{\mathrm{GRPO}}(\phi; \mathcal{D}_{\mathrm{sel}}) + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}(\phi; \mathcal{D}_{\mathrm{sel}}). \tag{13}$$

Here $\lambda_{\mathrm{KL}} = 0.001$ weights the explicit KL loss. The GRPO term uses the configured clipping and rollout importance-weight corrections. Current-actor forward and backward passes are performed on the selected training batch; replay scoring uses the cached quantities.

For K replay blocks with B prompt groups per block, n responses per group, and average response length L, there are $N = KBn$ candidate responses. D-ARL adds current-actor evaluation costing $O(NC_{\text{fwd}}(\theta, L))$, while GA-ARL scores cached tokens in $O(NL)$. Both methods select groups using the same top-M operation and then optimize the selected batch. Appendix F.3 details the scoring, selection, and storage costs.

5 Experiments

In this section, we evaluate the effectiveness and efficiency of GA-ARL from four aspects. **Experiment 1** compares reasoning performance across mathematical reasoning, logical reasoning, and code generation benchmarks. **Experiment 2** evaluates training efficiency in terms of wall-clock time and validation performance under the same time budget. **Experiment 3** conducts ablation studies on the key components of GA-ARL, including the replay buffer size and the advantage-weighted analytic target. **Experiment 4** analyzes the replay selection behavior of GA-ARL, including selected sample sources, replay selection overhead, and discrepancy to the analytic target.

Tasks and Datasets. We evaluate GA-ARL on five reasoning benchmarks covering mathematical, logical, and code reasoning. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and train on its official training split. For logical reasoning and code generation, we train on the corresponding logic and code subsets of Guru-RL-92K (Cheng et al., 2025), and evaluate on `ordering_puzzle`, `zebra_puzzle`, LiveCodeBench (Jain et al., 2024), and HumanEval (Chen et al., 2021) following their standard protocols. More dataset details are provided in Appendix E.

Experimental Setup. We use VERL (Sheng et al., 2025) as the unified training and evaluation framework, with Qwen3-1.7B and Qwen3-4B as base models. All methods share the same reward computation, advantage estimation and batch size for fair comparison. Unless otherwise specified, the batch size is 256 and the replay buffer length is 4. Asynchronous experiments are conducted on four NVIDIA A800-SXM4 GPUs (80GB memory each). More implementation details are provided in Appendix F.1.

Evaluation Metrics. We evaluate reasoning performance using accuracy for mathematical and logical reasoning tasks, and standard pass rate (pass@1) for code generation tasks. For training efficiency, we report validation performance under wall-clock training time and compare the total runtime required to complete the same number of training steps.

Baselines and Hyperparameters. We compare GA-ARL with ARLHF and D-ARL using a common VERL implementation and shared hyperparameters, including the backbone, reward computation, advantage estimator, and training budget. This comparison varies replay reuse and selection within a matched training pipeline. Hyperparameters and baseline details are provided in Appendix F.1.

- **ARLHF** (Noukhovitch et al., 2025a) adapts GRPO to asynchronous training by reusing samples from recent training steps with importance sampling correction.
- **D-ARL** (Bai et al., 2026) is a SOTA distribution-aware asynchronous RL method that maintains a replay buffer from multiple recent behavior policies and selects replay samples using a variance-aware criterion.

5.1 Experiment 1. Reasoning Performance Evaluation

In this subsection, we compare GA-ARL with asynchronous baselines on mathematical reasoning, code generation, and logical reasoning benchmarks under the same training steps. As shown in Table 1, GA-ARL achieves the best average performance on both Qwen3-1.7B and Qwen3-4B. For Qwen3-1.7B, GA-ARL improves the average accuracy by 6.1% over ARLHF and 3.1% over D-ARL. For Qwen3-4B, GA-ARL improves the average accuracy by 5.1% over ARLHF and 1.4% over D-ARL. Experiment 2 compares the corresponding wall-clock training costs. For more results, please refer to Appendix D.1.

Multi-seed validation. We compare GA-ARL and D-ARL on Qwen3-4B GSM8K over eight matched seeds (42–49), using a fixed 29-step budget. GA-ARL achieves $93.02 \pm 0.23\%$ final accuracy versus

Table 1: Accuracy comparison of GA-ARL and baseline methods using Qwen3-1.7B and Qwen3-4B. GA-ARL achieves the best average performance on both model sizes, demonstrating the effectiveness of advantage-weighted analytic replay selection.

Qwen3-1.7B	**Mathematical**	**Coding**		**Logical**		**Average**
	GSM8K	**HumanEval**	**LiveCodeBench**	**Ordering Puzzle**	**Zebra Puzzle**	
Method	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑
ARLHF	86.96	68.17	35.71	65.00	5.13	52.19
D-ARL	87.79	68.73	36.33	71.00	4.68	53.71
GA-ARL (Ours)	**89.23**	**71.17**	**37.35**	**72.00**	**7.05**	**55.36**
Qwen3-4B	**Mathematical**	**Coding**		**Logical**		**Average**
	GSM8K	**HumanEval**	**LiveCodeBench**	**Ordering Puzzle**	**Zebra Puzzle**	
Method	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%)	Acc (%) ↑	Acc (%) ↑
ARLHF	91.05	87.80	45.71	82.00	14.59	64.23
D-ARL	92.42	90.24	46.12	**84.00**	19.98	66.55
GA-ARL (Ours)	**92.80**	**92.07**	**49.41**	83.00	**20.10**	**67.48**

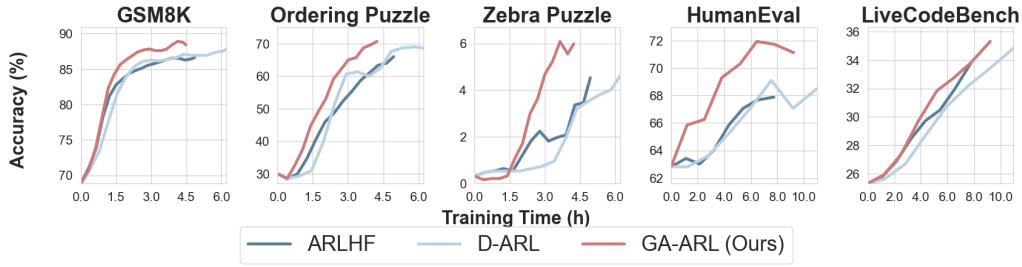


Figure 3: **Training efficiency comparison on Qwen3-1.7B.** We report validation accuracy over wall-clock training time. For each dataset, all methods are trained with the same number of training steps. GA-ARL generally reaches higher accuracy under the same time and reduces wall-clock training time compared with D-ARL.

$92.60 \pm 0.19\%$ for D-ARL, a mean paired gain of 0.42 percentage points (95% CI $[0.17, 0.67]$; two-sided paired t-test $p = 0.0053$). GA-ARL also reduces training time by $31.49 \pm 2.01\%$. Under the same per-seed time budget, D-ARL reaches $91.56 \pm 0.43\%$, yielding a larger paired gain of 1.47 points for GA-ARL ($p = 0.00012$). Standard deviations are across seeds; full results and the comparison protocol are provided in Appendix D.3.

We further evaluate code generation over eight matched seeds on Qwen3-4B LiveCodeBench and Qwen3-1.7B HumanEval. GA-ARL improves mean accuracy over D-ARL by 2.14 and 1.76 percentage points, respectively (paired t-test $p = 0.0060$ and $p = 0.00329$), while reducing training time by $35.36 \pm 1.15\%$ and $32.84 \pm 1.58\%$. Detailed results are provided in Appendix D.4.

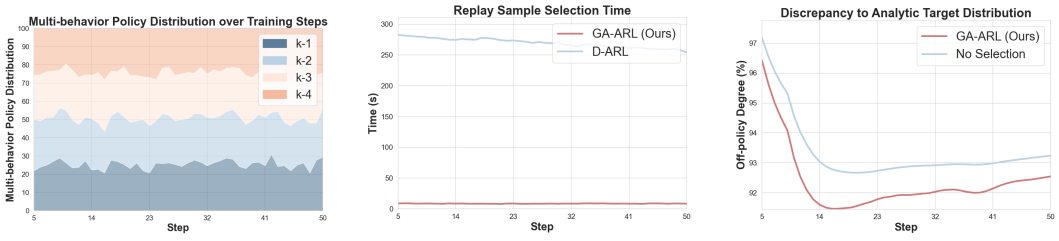
5.2 Experiment 2. Training Efficiency Evaluation

In this subsection, we compare GA-ARL with asynchronous baselines in terms of training efficiency on Qwen3-1.7B, including both wall-clock training time and validation performance under the same time budget. For each dataset, all methods are trained for the same number of training steps. As shown in Figure 3, GA-ARL generally reaches higher validation accuracy within the same training time across mathematical reasoning, logical reasoning, and code generation benchmarks. Compared with D-ARL, GA-ARL reduces the total wall-clock training time by an average of 25.3%, mainly because it avoids additional current-policy evaluation during replay scoring. Compared with ARLHF, GA-ARL achieves better validation accuracy under the same wall-clock time, with slightly higher total training costs on some long-response code tasks. For more results, please refer to Appendix D.2.

We also compare with synchronous GRPO on Qwen3-4B GSM8K using seed 42 and a one-epoch, 29-step budget. GA-ARL achieves 92.95% final accuracy versus 91.05% for GRPO. GRPO completes training sooner in this short-response setting; at its completion time, GA-ARL's latest validation

Table 2: **Ablation study of GA-ARL components on Qwen3-1.7B.** We evaluate the impact of replay buffer size and advantage-weighted analytic target selection. *GA-ARL w/ Reduced Buffer* uses a smaller replay buffer with $K = 2$, while *GA-ARL w/o A-weighted* removes the $|A|$-based weighting term from the analytic target distribution. Results show that both components contribute to the effectiveness of GA-ARL.

Datasets	GSM8K	HumanEval	LiveCodeBench	Ordering Puzzle	Zebra Puzzle	**Average**
Method	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑	Acc (%) ↑
GA-ARL (Ours)	**89.23**	**71.17**	**37.35**	**72.00**	**7.05**	**55.36**
GA-ARL w/ Reduced Buffer	88.02	68.07	34.49	70.00	5.77	53.27
GA-ARL w/o A-weighted	88.25	70.25	33.47	63.00	3.93	51.78



(a) Source distribution. (b) Replay sample selection time. (c) Discrepancy to analytic target.

Figure 4: **Analysis of GA-ARL on Qwen3-1.7B.** (a) GA-ARL selects samples from multiple recent behavior policies, showing the usefulness of maintaining a multi-policy replay buffer. (b) GA-ARL greatly reduces replay sample selection time compared with D-ARL by avoiding additional current-policy evaluation. (c) GA-ARL consistently selects samples with lower discrepancy to the analytic target distribution than no selection.

checkpoint reaches 91.89%. Thus, GA-ARL improves accuracy both at the final step and under the GRPO time budget. Appendix D.5 provides the results and protocol.

On Qwen3-4B LiveCodeBench with a maximum response length of $12{,}288$ tokens, GA-ARL achieves 62.65% test accuracy versus 58.38% for ARLHF under the same 66-hour wall-clock budget, an improvement of 4.27 percentage points. Appendix D.7 reports this long-response comparison.

5.3 Experiment 3. Ablation Study

In this subsection, we conduct ablation experiments to analyze the contribution of key components in GA-ARL. We compare GA-ARL with two representative variants: (i) *GA-ARL w/ Reduced Buffer*, which uses a smaller replay buffer and keeps samples from fewer recent behavior policies ($K = 2$), and (ii) *GA-ARL w/o A-weighted*, which removes the advantage-weighted term and selects samples according to the original analytic target distribution. As shown in Table 2, GA-ARL achieves the best performance on all available benchmarks. Reducing the replay buffer size leads to consistent performance drops, indicating that maintaining samples from more recent behavior policies provides more diverse and useful replay candidates. Moreover, removing the advantage-weighted target also results in degradation, especially on logical reasoning benchmarks, demonstrating that the $|A|$-weighted term is important for selecting samples with stronger learning signals.

We further test sensitivity to reward-induced advantage noise on Qwen3-4B GSM8K over eight matched seeds. With 1% and 5% binary reward flips before advantage estimation, GA-ARL exceeds D-ARL by 0.65 and 0.78 percentage points, respectively ($p = 0.0027$ and $p = 0.0059$). Relative to clean runs, GA-ARL's accuracy drops are smaller by 0.23 and 0.36 points (Holm-adjusted $p = 0.0206$ for both). Full results are reported in Appendix D.6.

5.4 Experiment 4. Analysis of Replay Sample Selection

In this subsection, we analyze the replay selection behavior of GA-ARL on Qwen3-1.7B. As shown in Figure 4, GA-ARL exhibits three key properties. First, Figure 4(a) shows that selected samples come from multiple recent behavior policies rather than being dominated by the most recent one, indicating that properly filtered historical rollouts can still provide useful training signals. Second, Figure 4(b) shows that GA-ARL substantially reduces replay selection time compared with D-ARL by avoiding additional current-policy evaluation and using cached reference-policy probabilities and advantage values. Third, Figure 4(c) shows that GA-ARL selects samples with lower discrepancy to the analytic target than no selection, confirming that its criterion identifies samples better aligned with the advantage-weighted analytic target.

6 Conclusion and Limitations and Broader impacts

Limitations and Broader Impacts. GA-ARL's replay selection depends on advantage quality. On Qwen3-4B GSM8K, 1% and 5% binary reward flips reduce accuracy, with smaller drops than D-ARL (Appendix D.6); sparse or systematically biased rewards remain important settings for further study. The gradient-based analysis uses a local-update approximation and comparable score-function norms, which may be less accurate for large updates or highly stale replay data. Evaluation covers five reasoning benchmarks with Qwen3-1.7B and Qwen3-4B; larger models, longer training runs, and more diverse tasks remain directions for future work. By reducing replay-scoring overhead, GA-ARL can make RL post-training more accessible, while also lowering the cost of optimizing models for harmful uses or biased reward objectives. Further discussion appears in Appendix A.

Conclusion. We proposed GA-ARL, a gradient-aware replay selection framework for asynchronous LLM post-training. An advantage-weighted analytic target guides replay scoring, which uses cached probabilities and advantages with a historical scaling estimate to select prompt groups without additional current-policy evaluation. Across five mathematical, logical, and code reasoning benchmarks, GA-ARL achieves relative average-accuracy gains of 3.1% and 1.4% over D-ARL on Qwen3-1.7B and Qwen3-4B, respectively, and reduces training time by 25.3% on average on Qwen3-1.7B. Eight-seed evaluations show accuracy gains and shorter training times; reward-noise tests and the long-response LiveCodeBench comparison further support the effectiveness of this replay selection approach. These results demonstrate that advantage-weighted analytic scoring can improve the accuracy–efficiency trade-off in asynchronous RL post-training.

Acknowledgments and Disclosure of Funding

This work was supported in part by the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0123102) and the National Natural Science Foundation of China (Grant No. U23A20388). We would like to thank all the anonymous reviewers for their insightful comments.

Competing interests. The authors declare no competing interests beyond the affiliations disclosed in this paper.

References

Bai, Y., Tong, X., Wang, J., Liu, H., Pan, L., Li, J., Wang, Z., HAO, J., Yuan, M., and Wu, F. D-arl: A distribution-matched asynchronous reinforcement learning framework for language reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, Seoul, South Korea, 2026. PMLR.

Chen, A., Li, A., Gong, B., Jiang, B., Fei, B., Yang, B., Shan, B., Yu, C., Wang, C., Zhu, C., et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter,

C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Cheng, Z., Hao, S., Liu, T., Zhou, F., Xie, Y., Yao, F., Bian, Y., Zhuang, Y., Dey, N., Zha, Y., Gu, Y., Zhou, K., Wang, Y., Li, Y., Fan, R., She, J., Gao, C., Saparov, A., Li, H., Killian, T. W., Yurochkin, M., Liu, Z., Xing, E. P., and Hu, Z. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective, 2025. URL https://arxiv.org/abs/2506.14965.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Fu, W., Gao, J., Shen, X., Zhu, C., Mei, Z., He, C., Xu, S., Wei, G., Mei, J., Wang, J., et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Hanna, J. P., Niekum, S., and Stone, P. Importance sampling in reinforcement learning with an estimated behavior policy. *Machine Learning*, 110(6):1267–1317, 2021.

Hilton, J., Cobbe, K., and Schulman, J. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Liang, X., Chen, C., Qiu, S., Wang, J., Wu, Y., Fu, Z., Chen, H., Wu, F., and Ye, J. ROPO: Robust preference optimization for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 37131–37161. PMLR, 2025a. URL https://proceedings.mlr.press/v267/liang25d.html.

Liang, X., Yang, L., Wang, J., Lu, Y., Wu, R., Chen, H., and Hao, J. Boosting multi-domain fine-tuning of large language models through evolving interactions between samples. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 37427–37441. PMLR, 2025b. URL https://proceedings.mlr.press/v267/liang25q.html.

Liang, X., Yang, L., Wang, J., Liu, R., Lu, Y., Zeng, J., Chen, H., Li, D., and Hao, J. Boosting multi-domain reasoning of LLMs via curvature-guided policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=R2EZtdHWJT.

Lin, B. Y., Bras, R. L., Richardson, K., Sabharwal, A., Poovendran, R., Clark, P., and Choi, Y. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

Lu, H., Liu, Z., Xiong, S., He, Y., Gao, W., Wu, Y., Wang, W., Liu, J., Li, Y., Zhao, H., et al. Part ii: Roll flash–accelerating rlvr and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

Meng, W., Zheng, Q., Pan, G., and Yin, Y. Off-policy proximal policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9162–9170, 2023.

Noukhovitch, M., Huang, S., Xhonneux, S., Hosseini, A., Agarwal, R., and Courville, A. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Noukhovitch, M., Huang, S., Xhonneux, S., Hosseini, A., Agarwal, R., and Courville, A. Faster, more efficient rlhf through off-policy asynchronous learning. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2025b.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafailov, Rafael, et al. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Roux, N. L., Bellemare, M. G., Lebensold, J., Bergeron, A., Greaves, J., Fréchette, A., Pelletier, C., Thibodeau-Laufer, E., Toth, S., and Work, S. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Team, Q. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Uehara, M., Shi, C., and Kallus, N. A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355*, 2022.

Zhang, K., Zuo, Y., He, B., Sun, Y., Liu, R., Jiang, C., Fan, Y., Tian, K., Jia, G., Li, P., et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025.

A Limitations and Broader Impacts

A.1 Limitations

GA-ARL relies on advantage estimates to construct the advantage-weighted analytic target distribution. On Qwen3-4B GSM8K, the eight-seed experiments with 1% and 5% binary reward flips show accuracy degradation for both GA-ARL and D-ARL, with smaller drops for GA-ARL (Appendix D.6). Sparse rewards and systematically biased rewards remain settings for further study. The gradient-based analysis uses a local update approximation and assumes comparable score-function norms across candidate samples. Large updates, highly stale replay data, or substantial variation in these norms may reduce the accuracy of this characterization. Our experiments cover five reasoning benchmarks with Qwen3-1.7B and Qwen3-4B. Larger models, longer training runs, and more diverse tasks remain directions for future work.

A.2 Broader Impacts

This work studies efficient asynchronous reinforcement learning for LLM post-training. Potential positive impacts include reducing the computational cost of RL post-training, improving hardware utilization, and making reasoning-oriented model improvement more accessible to researchers with limited computational resources. By avoiding additional current-policy evaluation during replay selection, GA-ARL may also reduce training time and energy consumption compared with more expensive asynchronous replay selection methods.

At the same time, improving the efficiency of LLM post-training may lower the cost of optimizing models for both beneficial and harmful downstream purposes. More efficient RL training could be used to improve models that generate misleading content, automate low-quality information production, or optimize behavior according to biased or unsafe reward signals. In addition, if reward models or advantage estimates encode undesirable preferences, replay selection may amplify these signals during training.

B Theoretical Analysis

B.1 Proof of Theorem 4.1

For a fixed state s, consider the KL-regularized state-level objective

$$\mathcal{F}_s(\pi) = \sum_a \pi(a|s)A(s,a) - \beta KL\left(\pi(\cdot|s)\|\pi_{\text{ref}}(\cdot|s)\right), \tag{14}$$

where the policy distribution satisfies the normalization constraint

$$\sum_a \pi(a|s) = 1. \tag{15}$$

Expanding the KL term, we have

$$\mathcal{F}_s(\pi) = \sum_a \pi(a|s)A(s,a) - \beta \sum_a \pi(a|s) \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)}. \tag{16}$$

To maximize $\mathcal{F}_s(\pi)$ under the normalization constraint, we introduce the Lagrangian

$$\mathcal{L}(\pi, \lambda) = \sum_a \pi(a|s)A(s,a) - \beta \sum_a \pi(a|s) \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} + \lambda \left(\sum_a \pi(a|s) - 1 \right). \tag{17}$$

Taking the derivative with respect to $\pi(a|s)$ and setting it to zero gives

$$A(s,a) - \beta \left(\log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} + 1 \right) + \lambda = 0. \tag{18}$$

Rearranging terms, we obtain

$$\log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} = \frac{A(s,a)}{\beta} + \frac{\lambda}{\beta} - 1. \tag{19}$$

Therefore,

$$\pi(a|s) = \pi_{\text{ref}}(a|s) \exp\left(\frac{A(s,a)}{\beta}\right) \exp\left(\frac{\lambda}{\beta} - 1\right). \tag{20}$$

The factor $\exp(\frac{\lambda}{\beta} - 1)$ is independent of a and is determined by the normalization constraint. Let

$$Z(s) = \sum_{a'} \pi_{\text{ref}}(a'|s) \exp\left(\frac{A(s,a')}{\beta}\right). \tag{21}$$

Then the normalized optimal distribution is

$$\pi^*(a|s) = \frac{\pi_{\text{ref}}(a|s) \exp(A(s,a)/\beta)}{Z(s)}. \tag{22}$$

We next show that this distribution characterizes the preferred policy direction induced by the KL-regularized objective. From the definition of π^*, we have

$$\log \pi^*(a|s) = \log \pi_{\text{ref}}(a|s) + \frac{A(s,a)}{\beta} - \log Z(s). \tag{23}$$

Thus,

$$A(s,a) = \beta \log \frac{\pi^*(a|s)}{\pi_{\text{ref}}(a|s)} + \beta \log Z(s). \tag{24}$$

Substituting this identity into $\mathcal{F}_s(\pi)$ yields

$$\begin{aligned}
\mathcal{F}_s(\pi) &= \sum_a \pi(a|s) \left[\beta \log \frac{\pi^*(a|s)}{\pi_{\text{ref}}(a|s)} + \beta \log Z(s)\right] - \beta \sum_a \pi(a|s) \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} \\
&= \beta \log Z(s) + \beta \sum_a \pi(a|s) \log \frac{\pi^*(a|s)}{\pi_{\text{ref}}(a|s)} - \beta \sum_a \pi(a|s) \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} \\
&= \beta \log Z(s) - \beta \sum_a \pi(a|s) \log \frac{\pi(a|s)}{\pi^*(a|s)} \\
&= \beta \log Z(s) - \beta KL\left(\pi(\cdot|s) \| \pi^*(\cdot|s)\right).
\end{aligned} \tag{25}$$

Since

$$KL\left(\pi(\cdot|s) \| \pi^*(\cdot|s)\right) \geq 0, \tag{26}$$

we have

$$\mathcal{F}_s(\pi) \leq \beta \log Z(s), \tag{27}$$

with equality if and only if

$$\pi(\cdot|s) = \pi^*(\cdot|s). \tag{28}$$

Therefore, maximizing the KL-regularized state-level objective is equivalent to minimizing the forward KL divergence from the updated policy to π^*. This proves that π^* characterizes the policy direction preferred by the current KL-regularized update.

B.2 Proof of Proposition 4.2

For a replay sample (s, a) generated by the behavior policy π_{θ_k}, the policy-gradient part of the sample loss can be written as

$$\ell_{\text{pg}}(\theta; s, a) = -\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A(s,a). \tag{29}$$

Taking the gradient with respect to θ, we obtain

$$\begin{aligned}
\nabla_\theta \ell_{\text{pg}}(\theta; s, a) &= -A(s,a) \nabla_\theta \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \\
&= -\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A(s,a) \nabla_\theta \log \pi_\theta(a|s).
\end{aligned} \tag{30}$$

Therefore, the update direction contributed by this sample is

$$-\nabla_\theta \ell_{\mathrm{pg}}(\theta; s, a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A(s,a) \nabla_\theta \log \pi_\theta(a|s). \tag{31}$$

We denote this sample-level update contribution as

$$g(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A(s,a) \nabla_\theta \log \pi_\theta(a|s). \tag{32}$$

Its magnitude satisfies

$$\|g(s,a)\| = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} |A(s,a)| \, \|\nabla_\theta \log \pi_\theta(a|s)\| \, . \tag{33}$$

Under a local update approximation, the updated policy is encouraged to move toward the analytic target policy induced by the KL-regularized objective. Therefore, for the purpose of characterizing the target-prioritized contribution of a replay sample, we approximate

$$\pi_\theta(a|s) \approx \pi^*(a|s). \tag{34}$$

In addition, we assume that the score-function norm changes mildly across candidate samples, i.e.,

$$\|\nabla_\theta \log \pi_\theta(a|s)\| \approx C, \tag{35}$$

where C is approximately constant for samples being compared.

Substituting these approximations into the magnitude of $g(s,a)$ gives

$$\|g(s,a)\| \approx C \frac{\pi^*(a|s)}{\pi_{\theta_k}(a|s)} |A(s,a)|. \tag{36}$$

This expression combines the behavior-policy importance factor $1/\pi_{\theta_k}(a|s)$ with the target-side factor used to construct the replay score:

$$\pi^*(a|s)|A(s,a)|. \tag{37}$$

This shows that $\pi^*(a|s)$ reflects whether the action is aligned with the KL-regularized target policy direction, while $|A(s,a)|$ reflects the strength of the learning signal. Hence, under the local update approximation, the target-prioritized gradient strength of a replay sample is proportional to

$$\pi^*(a|s)|A(s,a)|. \tag{38}$$

This completes the proof.

C More Details on the Background and Related Work

C.1 Off-policy Correction with Importance Sampling

In asynchronous RL, replay samples may be generated by a stale behavior policy π_{θ_k} rather than the current policy π_θ. Importance sampling is therefore used to correct this distribution mismatch. For a trajectory τ, the correction coefficient is

$$w(\tau) = \frac{\pi_\theta(\tau)}{\pi_{\theta_k}(\tau)}. \tag{39}$$

For language model post-training, where $\tau = (a_0, \ldots, a_{N-1})$, this ratio can be factorized into token-level probabilities:

$$w(\tau) = \prod_{i=0}^{N-1} \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)}. \tag{40}$$

With this correction, local policy optimization still keeps its original update ratio between the optimized policy π_ϕ and the current policy π_θ. Thus, the importance-corrected token-level policy-gradient term can be written as

$$\mathbb{E}_{s,a\sim\pi_{\theta_k}} \left[w(s,a) \frac{\pi_\phi(a|s)}{\pi_\theta(a|s)} A(s,a) \right], \tag{41}$$

where

$$w(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \tag{42}$$

is the token-level importance sampling coefficient that corrects the mismatch between the stale behavior policy and the current policy. In practice, large deviations between π_{θ_k} and π_θ can make w high-variance, motivating clipping, proximal constraints, or replay sample filtering in asynchronous RL.

C.2 Details of D-ARL

D-ARL is a replay-based asynchronous RL method that mitigates the instability caused by stale off-policy samples through replay sample selection. It maintains a replay buffer containing samples generated by the previous K behavior policies, and selects samples that are closer to a minimum-variance behavior distribution for policy optimization.

Specifically, let π_p denote the target policy and π_q denote the behavior policy. Given trajectories sampled from π_q, the importance sampling estimator is

$$\hat{J}_{\pi_q} = \frac{1}{n}\sum_{i=1}^{n} \frac{\pi_p(\tau_i)}{\pi_q(\tau_i)} R(\tau_i). \tag{43}$$

D-ARL derives the behavior distribution that minimizes the variance of this estimator:

$$\pi_q^*(\tau) = \frac{\pi_p(\tau)R(\tau)}{\mathbb{E}_{\tau\sim\pi_p}[R(\tau)]}. \tag{44}$$

Since the denominator is generally unavailable, D-ARL approximates it using historical return statistics. Based on this target distribution, D-ARL selects replay samples whose distributions better match π_q^*, thereby reducing the variance of off-policy optimization and improving training stability.

However, computing $\pi_q^*(\tau)$ requires evaluating candidate replay samples under the current target policy π_p. In LLM post-training, this introduces substantial overhead when responses are long or the replay buffer is large. In addition, D-ARL primarily focuses on variance reduction, which may not fully capture whether a replay sample is aligned with the desired update direction of the current RL objective.

C.3 Gradient-informed Data Selection and Policy Optimization

EVIC (Liang et al., 2025b) estimates interactions between training samples through gradients and updates the training curriculum to select samples that benefit the overall dataset. CGPO (Liang et al., 2026) uses cross-domain curvature information to promote gradient alignment during multi-domain reinforcement learning. GA-ARL uses an advantage-weighted analytic target to select replay samples for asynchronous policy optimization.

D Additional Results

D.1 More Results for Reasoning Performance Comparative Evaluation

Figure 5 and Figure 6 provide additional reasoning performance comparisons of GA-ARL against asynchronous baselines. As shown in Figure 5, on Qwen3-1.7B, GA-ARL consistently achieves stronger validation performance across mathematical reasoning, logical reasoning, and code generation benchmarks. In particular, GA-ARL reaches higher accuracy earlier on GSM8K, Ordering Puzzle, Zebra Puzzle, HumanEval, and LiveCodeBench, indicating better sample efficiency under the same number of policy updates. Similar trends can be observed on Qwen3-4B in Figure 6, where GA-ARL obtains the best or comparable-best performance across most benchmarks and shows faster improvement during training. On Qwen3-4B Ordering Puzzle, GA-ARL achieves lower accuracy than D-ARL at the same training-step budget and competitive accuracy under the same wall-clock budget (Appendix D.2).

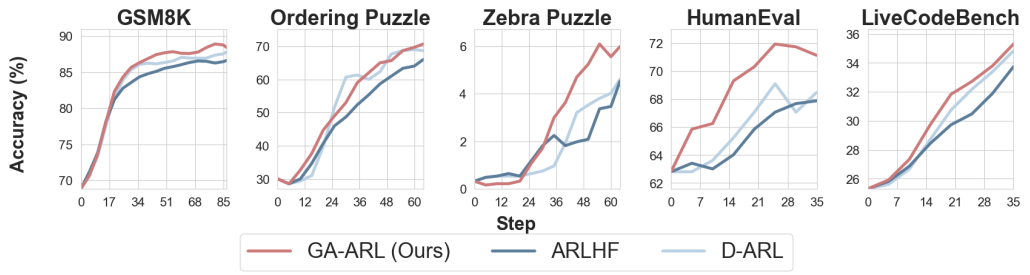


Figure 5: **Reasoning performance comparison on Qwen3-1.7B.** We report validation accuracy over training steps on mathematical reasoning, logical reasoning, and code generation benchmarks. GA-ARL consistently achieves better or comparable performance than ARLHF and D-ARL, showing stronger sample efficiency across different tasks.

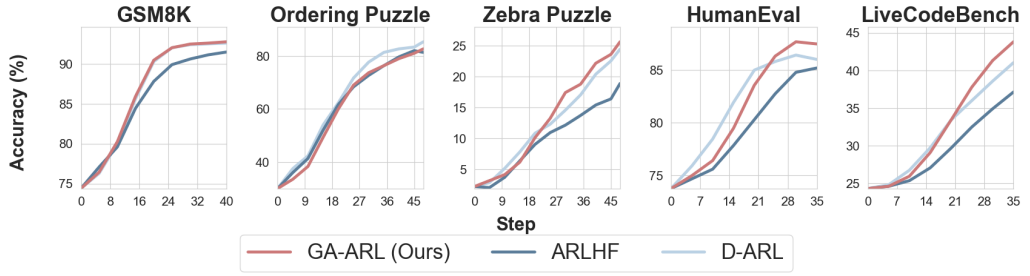


Figure 6: **Reasoning performance comparison on Qwen3-4B.** We report validation accuracy over training steps on mathematical reasoning, logical reasoning, and code generation benchmarks. GA-ARL generally reaches higher accuracy with fewer training steps, demonstrating the effectiveness of advantage-weighted analytic replay selection on larger models.

D.2 More Results for Training Efficiency Comparative Evaluation

Figure 7 provides additional training efficiency comparisons on Qwen3-4B under wall-clock training time. As shown in the figure, GA-ARL generally reaches higher accuracy earlier than ARLHF and D-ARL across mathematical reasoning, logical reasoning, and code generation benchmarks. In particular, GA-ARL shows clear efficiency advantages on GSM8K, Zebra Puzzle, HumanEval, and LiveCodeBench, where it achieves stronger validation performance within fewer training hours. On Ordering Puzzle, GA-ARL reaches competitive accuracy under the same wall-clock budget. Replay scoring reuses cached quantities and avoids evaluating the current policy on all candidates.

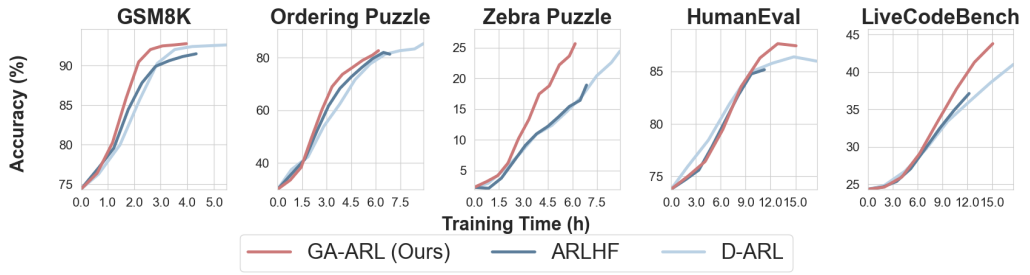


Figure 7: **Training efficiency comparison on Qwen3-4B.** We report validation accuracy over wall-clock training time on mathematical reasoning, logical reasoning, and code generation benchmarks. GA-ARL generally reaches higher accuracy within fewer training hours, showing better practical training efficiency than ARLHF and D-ARL.

D.3 Multi-seed Validation on GSM8K

We evaluate Qwen3-4B on GSM8K using eight matched seeds, 42–49. Each run uses a fixed budget of 29 training steps. Within each seed, GA-ARL and D-ARL share the same backbone, data split, reward function, rollout configuration, optimization hyperparameters, training steps, and hardware; they differ in replay selection.

For the matched-time comparison, let $T_{\mathrm{GA},j}$ and $T_{\mathrm{D},j}$ denote the total training times for seed j. We use D-ARL's latest logged validation checkpoint whose timestamp does not exceed $T_{\mathrm{GA},j}$, without interpolation, and compare it with GA-ARL's final accuracy. The per-seed time reduction is $100(1 - T_{\mathrm{GA},j}/T_{\mathrm{D},j})\%$; we report the mean and sample standard deviation of these reductions. Table 3 lists the seed-level results.

Table 3: Eight-seed Qwen3-4B GSM8K validation (29 training steps). Accuracy is in percent. D-ARL at GA time uses the latest checkpoint within the corresponding GA-ARL training-time budget. GA-ARL at that budget equals its final accuracy. All $\pm$ values are sample standard deviations across seeds.

Seed	D-ARL final	GA-ARL final	D-ARL at GA time	Time reduction
42	92.72	92.95	92.32	28.37%
43	92.42	93.25	91.56	29.48%
44	92.87	93.33	90.80	30.37%
45	92.55	92.75	91.44	31.27%
46	92.31	92.92	91.82	31.94%
47	92.76	92.71	91.31	32.83%
48	92.48	93.23	91.67	33.50%
49	92.69	93.03	91.52	34.17%
Mean $\pm$ std	92.60 ± 0.19	93.02 ± 0.23	91.56 ± 0.43	$31.49 \pm 2.01\%$

For each comparison, we compute the within-seed accuracy difference $\Delta_j = \mathrm{Acc}_{\mathrm{GA},j} - \mathrm{Acc}_{\mathrm{D},j}$ and use a two-sided paired t-test with seven degrees of freedom. The 95% confidence interval for the mean difference is $\bar{\Delta} \pm t_{0.975,7} s_\Delta / \sqrt{8}$, where s_Δ is the sample standard deviation of the eight differences. Table 4 reports both comparisons.

Table 4: Paired comparisons on the eight-seed GSM8K validation. Accuracy is reported as mean $\pm$ sample standard deviation (%). Gains and confidence intervals are in percentage points.

Comparison	D-ARL	GA-ARL	Gain	95% CI	p
Final	92.60 ± 0.19	93.02 ± 0.23	$+0.42$	$[0.17, 0.67]$	0.0053
Matched-time	91.56 ± 0.43	93.02 ± 0.23	$+1.47$	$[1.01, 1.92]$	0.00012

GA-ARL achieves higher mean final accuracy while reducing training time, with a larger accuracy gain under a matched time budget.

D.4 Multi-seed Validation on Code Generation

We compare GA-ARL with D-ARL on Qwen3-4B LiveCodeBench and Qwen3-1.7B HumanEval using eight matched seeds, 42–49. For each setting and seed, both methods share the same data split, rollout configuration, reward function, optimization hyperparameters, training steps, and hardware. Tables 5 and 6 report the per-seed accuracy and training-time reduction, with sample standard deviations across seeds. Time reductions are calculated relative to D-ARL for each seed and then averaged.

Table 5: Eight-seed results on Qwen3-4B LiveCodeBench. Accuracy and training-time reduction are reported in percent; $\pm$ denotes the sample standard deviation across seeds.

Seed	D-ARL	GA-ARL	Time reduction
42	46.12	49.41	33.57%
43	46.86	47.66	34.21%
44	45.69	49.69	34.72%
45	46.47	46.07	35.23%
46	45.88	49.48	35.62%
47	46.75	47.75	36.13%
48	46.20	49.00	36.51%
49	45.96	47.96	36.89%
Mean $\pm$ std	46.24 ± 0.42	48.38 ± 1.24	$35.36 \pm 1.15\%$

Table 6: Eight-seed results on Qwen3-1.7B HumanEval. Accuracy and training-time reduction are reported in percent; $\pm$ denotes the sample standard deviation across seeds.

Seed	D-ARL	GA-ARL	Time reduction
42	68.02	71.29	30.38%
43	69.14	69.30	31.26%
44	68.84	71.32	31.96%
45	68.52	69.57	32.66%
46	68.73	70.31	33.19%
47	68.24	71.38	33.89%
48	68.24	70.02	34.42%
49	69.54	70.16	34.94%
Mean $\pm$ std	68.66 ± 0.51	70.42 ± 0.82	$32.84 \pm 1.58\%$

We use the paired statistical procedure described in Appendix D.3 to compare accuracy across matched seeds. On LiveCodeBench, GA-ARL achieves $48.38 \pm 1.24\%$ versus $46.24 \pm 0.42\%$ for D-ARL, with a mean paired gain of 2.14 percentage points (95% CI $[0.84, 3.44]$, $p = 0.0060$). On HumanEval, GA-ARL achieves $70.42 \pm 0.82\%$ versus $68.66 \pm 0.51\%$, with a mean paired gain of 1.76 points (95% CI $[0.81, 2.71]$, $p = 0.00329$). Together with the training-time reductions of $35.36 \pm 1.15\%$ and $32.84 \pm 1.58\%$, these results demonstrate improved code-generation performance and training efficiency on both settings.

D.5 Comparison with Synchronous GRPO

We compare synchronous GRPO with GA-ARL on Qwen3-4B GSM8K using seed 42, one epoch, and 29 training steps. Both methods use the same backbone, data split, reward function, and optimization configuration. GRPO provides a synchronous reference with the same group-based advantage estimator and a closely matched actor objective.

Table 7 reports final accuracy and accuracy at GRPO's completion time. For GA-ARL at this time budget, we use its latest logged validation checkpoint before the cutoff, without interpolation. Synchronous GRPO completes the 29 steps sooner in this short-response setting, where the rollout latency available to overlap with policy optimization is limited. GA-ARL nevertheless achieves higher accuracy at both the final step and the matched-time cutoff.

Table 7: Synchronous GRPO comparison on Qwen3-4B GSM8K (seed 42, one epoch, 29 steps). Accuracy is in percent. The matched-time budget is GRPO's completion time.

Method	Final accuracy	Accuracy at GRPO time
GRPO	91.05	91.05
GA-ARL	92.95	91.89
Gain (percentage points)	$+1.90$	$+0.84$

D.6 Robustness to Reward Noise

ROPO (Liang et al., 2025a) improves robustness to noisy preferences through a robust loss and iterative noise filtering.

We evaluate robustness to reward-induced advantage noise on Qwen3-4B GSM8K using eight matched seeds, 42–49. Each response-level binary outcome reward is independently flipped with probability 0.01 or 0.05 before advantage estimation. The perturbed rewards are used consistently for advantage computation, replay selection, and policy optimization. Clean runs use a flip probability of zero. GA-ARL and D-ARL share the same data, rollout and optimization settings, training steps, and hardware within each matched seed.

Table 8 gives the complete seed-level accuracies. Table 9 summarizes their means and sample standard deviations, together with two-sided paired t-tests and 95% confidence intervals calculated using the procedure in Appendix D.3.

Table 8: Seed-level accuracy (%) under reward corruption on Qwen3-4B GSM8K. Column labels within each method denote reward flip probabilities.

	D-ARL			GA-ARL		
Seed	Clean	1%	5%	Clean	1%	5%
42	92.72	92.22	90.50	92.95	92.80	90.83
43	92.42	91.94	89.43	93.25	93.10	90.98
44	92.87	92.42	90.14	93.33	92.88	90.55
45	92.55	91.96	89.30	92.75	92.38	89.79
46	92.31	91.52	89.09	92.92	92.16	90.49
47	92.76	92.16	90.03	92.71	92.20	90.15
48	92.48	91.67	89.48	93.23	92.98	90.85
49	92.69	92.19	89.86	93.03	92.80	90.42

Table 9: Accuracy under reward noise, reported as mean $\pm$ sample standard deviation (%). Gains and 95% confidence intervals are for GA-ARL minus D-ARL, in percentage points; p-values are from two-sided paired t-tests.

Flip probability	D-ARL	GA-ARL	Gain	95% CI	p
0	92.60 ± 0.19	93.02 ± 0.23	$+0.42$	$[0.17, 0.67]$	0.0053
1%	92.01 ± 0.30	92.66 ± 0.36	$+0.65$	$[0.31, 0.99]$	0.0027
5%	89.73 ± 0.48	90.51 ± 0.40	$+0.78$	$[0.31, 1.25]$	0.0059

To compare degradation, we compute each method's within-seed change from its clean run, $\delta_{m,j} = \mathrm{Acc}_{m,j}^{\mathrm{noisy}} - \mathrm{Acc}_{m,j}^{\mathrm{clean}}$, and compare $\delta_{\mathrm{GA},j} - \delta_{\mathrm{D},j}$ across seeds. A positive difference indicates a smaller drop for GA-ARL. Table 10 reports the mean changes, paired differences, and 95% confidence intervals. We apply Holm correction to the two paired tests of degradation differences at the 1% and 5% noise levels.

Table 10: Accuracy changes relative to clean runs (percentage points). Method columns show mean $\pm$ sample standard deviation. Differences are GA-ARL minus D-ARL changes; p_{Holm} is adjusted over the two noise-level comparisons.

Flip probability	D-ARL change	GA-ARL change	Difference	95% CI	p_{Holm}
1%	-0.59 ± 0.14	-0.36 ± 0.21	$+0.23$	$[0.07, 0.39]$	0.0206
5%	-2.87 ± 0.33	-2.51 ± 0.27	$+0.36$	$[0.10, 0.62]$	0.0206

Reward corruption reduces accuracy for both methods. GA-ARL maintains higher accuracy at both noise levels and shows smaller drops from its clean runs, demonstrating resilience to reward-induced advantage noise.

D.7 Long-response Evaluation on LiveCodeBench

We evaluate GA-ARL and ARLHF on Qwen3-4B LiveCodeBench with a maximum response length of $12{,}288$ tokens. The comparison uses a common wall-clock budget of 66 hours. Longer responses increase rollout generation cost and the amount of token-level data available for replay selection.

Table 11 reports test accuracy within this budget. GA-ARL reaches 62.65%, compared with 58.38% for ARLHF, yielding a 4.27-point improvement.

Table 11: Long-response Qwen3-4B LiveCodeBench evaluation with a maximum response length of $12{,}288$ tokens. Both methods are evaluated under a common 66-hour wall-clock budget.

Method	Test accuracy (%)
ARLHF	58.38
GA-ARL	62.65
Gain (percentage points)	$+4.27$

E Details of Assets Used in This Paper

In our experiments, we evaluate GA-ARL on mathematical reasoning, code generation, and logical reasoning benchmarks. The training data are constructed from the corresponding training split of each task or from domain-specific subsets of Guru-RL-92K.

E.1 Mathematics-Related Dataset

- **GSM8K** (Cobbe et al., 2021) is a widely used benchmark for grade-school mathematical reasoning. It consists of arithmetic word problems that require multi-step reasoning and numerical computation. In our experiments, we use a subset of the official GSM8K training set for RL training and evaluate models on the GSM8K test set.

E.2 Code-Related Datasets

- **LiveCodeBench** (Jain et al., 2024) is a code generation benchmark constructed from recent programming problems. It evaluates the ability of language models to generate correct programs under realistic coding scenarios. In our experiments, models are trained using code-domain samples from Guru-RL-92K and evaluated on LiveCodeBench following its standard evaluation protocol.
- **HumanEval** (Chen et al., 2021) is a widely adopted benchmark for evaluating functional correctness in code generation. It contains Python programming problems paired with unit tests, which are used to verify whether generated solutions satisfy the required specifications. We use code-domain samples from Guru-RL-92K for training and evaluate models on the HumanEval test set.

E.3 Logic-Related Datasets

- **Ordering Puzzle** (Cheng et al., 2025) evaluates logical reasoning ability through ordering and constraint-satisfaction problems. These tasks require models to infer a valid ordering of entities under a set of given constraints. We use logic-domain samples from Guru-RL-92K for training and evaluate models on the Ordering Puzzle test set from the official Guru evaluation release.
- **Zebra Puzzle** (Lin et al., 2025) is a logical reasoning benchmark based on constraint satisfaction. Models are required to infer hidden assignments by combining multiple clues. We use logic-domain samples from Guru-RL-92K for training and evaluate models on the Zebra Puzzle test set from the official Guru evaluation release.

E.4 Training Data and Splits

Table 12 summarizes the sizes of the training and test sets used in our experiments. For GSM8K, we use a subset of its corresponding official training set. For LiveCodeBench, HumanEval, Ordering Puzzle, and Zebra Puzzle, we use the corresponding domain-specific subsets from **Guru-RL-92K** (Cheng et al., 2025). Specifically, we select approximately 3K code samples for LiveCodeBench and HumanEval, and approximately 4K logic samples for Ordering Puzzle and Zebra Puzzle. All methods are trained and evaluated using the same data splits for fair comparison.

Table 12: The sizes of the training and test sets for each dataset.

Benchmark	Training Set	Test Set
GSM8K	5K	1,319
LiveCodeBench	3K	510
HumanEval	3K	164
Ordering Puzzle	4K	100
Zebra Puzzle	4K	200

E.5 Licenses and Terms of Use for Existing Assets

Our experiments use existing public datasets, benchmarks, model checkpoints, and training/evaluation frameworks. We cite the original papers or repositories for all assets and use them only for research and evaluation purposes, following their released licenses and terms of use.

- **GSM8K** (Cobbe et al., 2021) is released under the MIT License. We use a subset of its official training split and evaluate on its official test set.
- **Guru-RL-92K** (Cheng et al., 2025) is released under the MIT License. We use its code-domain and logic-domain subsets to construct training data for LiveCodeBench, HumanEval, Ordering Puzzle, and Zebra Puzzle experiments.
- **LiveCodeBench** (Jain et al., 2024) is released under the MIT License. We use it as an evaluation benchmark and follow its standard evaluation protocol.
- **HumanEval** (Chen et al., 2021) is released under the MIT License. We use its official benchmark problems and unit tests for code generation evaluation.
- **Ordering Puzzle and Zebra Puzzle** (Cheng et al., 2025; Lin et al., 2025) are evaluated using the official Guru evaluation release, with 100 Ordering Puzzle examples and 200 Zebra Puzzle examples. These files are distributed in the Guru dataset repository, whose dataset card declares the MIT License. Guru dataset card.
- **Qwen3-1.7B and Qwen3-4B** (Team, 2025) are used as the base model checkpoints in our experiments. They are released under the Apache License 2.0, and we follow the official license and usage terms of the released checkpoints.
- **VERL** (Sheng et al., 2025) is used as the unified RL training and evaluation framework. It is released under the Apache License 2.0, and we follow its open-source license and usage terms.

Our released implementation builds on VERL and retains its upstream copyright notices and Apache 2.0 license; bundled components retain their respective notices. Model checkpoints and the exact paper data splits are supplied separately by users, as described in the repository's configuration guide.

F Details of Methods and Experimental Settings

F.1 Details of Experimental Setup

In this section, we provide additional details about the implementation of baselines, training configurations, and hyperparameter settings used in our experiments.

Code Availability. The GA-ARL implementation is publicly available at `https://github.com/MIRALab-USTC/GA-ARL`. The repository provides installation instructions, a configuration guide, and an index of training scripts for the reasoning tasks, replay ablations, and long-response LiveCodeBench evaluation. The method is implemented in `recipe/async_replay`. The configuration guide describes the model and data paths, input data schema, GPU allocation, and training parameters; the scripts run periodic validation during training. Dataset sources and experimental settings are described in Appendix E and Table 13.

Compute Resources. All asynchronous experiments use four NVIDIA A800-SXM4 GPUs with 80GB memory each, with two GPUs assigned to rollout generation and two to policy training. This allocation is shared by the main experiments, the additional multi-seed and reward-noise evaluations, and the long-response LiveCodeBench comparison. The long-response comparison uses a common 66-hour wall-clock budget, corresponding to an allocation of 264 GPU-hours per method (Appendix D.7).

Implementation Details of the Baselines. ARLHF, D-ARL, and GA-ARL are reimplemented in the same VERL codebase. Each matched comparison uses the same backbone, data split, rollout configuration, reward function, advantage estimator, optimizer, batch size, and training budget. The common configuration is used directly for all three methods, with no method-specific hyperparameter sweeps. Their replay reuse and selection mechanisms are described below. Synchronous GRPO provides an additional reference in Appendix D.5.

ARLHF (Noukhovitch et al., 2025a) is an asynchronous RL baseline that updates the policy using samples generated by a recent behavior policy. Its implementation follows the standard asynchronous setting in prior work, where replay samples are directly used for policy optimization under the same GRPO-style objective.

D-ARL (Bai et al., 2026) is a replay-based asynchronous RL method that maintains samples from multiple recent behavior policies and selects replay data according to a variance-aware criterion. In our experiments, we set the replay buffer length to $K = 4$ and use a sample warmup of 5 training steps before replay selection is activated.

GA-ARL is our proposed method. It uses the same replay buffer length as D-ARL, i.e., $K = 4$, and applies replay selection according to the discrepancy between replay samples and the advantage-weighted analytic target distribution. The sample warmup is also set to 5 steps for consistency.

Hyperparameters. Table 13 summarizes the default hyperparameters for the main experiments, including model configuration, replay buffer settings, rollout sampling, and optimization-related parameters. Experiment-specific settings are provided in the corresponding appendix sections.

Table 13: **Default hyperparameter settings for the main experiments.** We report the main model, rollout, replay buffer, optimization configurations, and training epochs.

Parameter	Value
Model Kwargs	
Max prompt length	1024
Max response length for GSM8K and logic-related tasks	2048
Max response length for code-related tasks	4096
Train Kwargs	
Replay buffer length	4
Train data batch size	256
Optimizer learning rate	1×10^{-6}
Replay-scoring temperature β_{target}	1
Actor KL-loss coefficient λ_{KL}	0.001
Clipping threshold	0.2
Clipping threshold c	3.0
Rollout.n	4
Sample warmup	5
Training Epochs	
Qwen3-1.7B on GSM8K	4
Qwen3-1.7B on code-related tasks	3
Qwen3-1.7B on logic-related tasks	3
Qwen3-4B on GSM8K	2
Qwen3-4B on code-related tasks	3
Qwen3-4B on logic-related tasks	3

F.2 Sampled-token Replay Scoring

Responses are grouped by their rollout prompt identifiers. For group g, let μ_g and σ_g be the mean and sample standard deviation of the response rewards. With group standard-deviation normalization enabled, the response advantage and its token broadcast are

$$\widehat{A}_\tau = \frac{R_\tau - \mu_g}{\sigma_g + \epsilon_A}, \qquad A_{\tau i} = m_{\tau i}\widehat{A}_\tau, \qquad \epsilon_A = 10^{-6}. \tag{45}$$

Advantages are computed from stored rewards when the replay buffer is scored and recomputed for the selected training groups before the actor update.

After actor update h, the masked mean token advantage of the selected batch is

$$\bar{A}_h = \frac{\sum_{\tau \in \mathcal{D}_{\text{sel},h}} \sum_i m_{\tau i} A_{\tau i}}{\sum_{\tau \in \mathcal{D}_{\text{sel},h}} \sum_i m_{\tau i}}. \tag{46}$$

Before scoring at step t, it averages the recorded values $\bar{A}_h \exp(\bar{A}_h)$ over all preceding actor updates, including warmup:

$$\widehat{z}_t = \frac{1}{|\mathcal{H}_t|} \sum_{h \in \mathcal{H}_t} \bar{A}_h \exp(\bar{A}_h), \tag{47}$$

where $\mathcal{H}_t$ indexes those updates. The same $\widehat{z}_t$ is used for all candidate tokens at step t.

The advantage-weighted log score is

$$v_{\tau i} = \ell_{\tau i}^{\text{ref}} + A_{\tau i} + \log \left| \frac{A_{\tau i} + \epsilon}{\widehat{z}_t + \epsilon} \right|, \qquad \epsilon = 10^{-8}, \tag{48}$$

which gives Eq. (9) at $\beta_{\text{target}} = 1$. Token discrepancies are evaluated as $1 - \exp(\min(v_{\tau i}, \ell_{\tau i}^{\text{b}}) - \max(v_{\tau i}, \ell_{\tau i}^{\text{b}}))$, masked, and averaged first over each response and then over its prompt group. The M groups with the smallest scores are selected with all their responses. Main GA-ARL runs score the full valid response; the training batch retains the complete selected responses.

F.3 Replay Discrepancy and Computational Cost

The max-normalized discrepancy can be written as $d(p, q) = 1 - \min(p, q)/\max(p, q)$ for positive scores. For example, $p = 10^{-10}$ and $q = 10^{-12}$ give $d(p, q) = 0.99$, capturing their relative disagreement despite an absolute difference below 10^{-10}. Symmetry treats either direction of mismatch consistently, and the bounded scale supports averaging across tokens and responses. The log-space calculation evaluates this relative difference using cached sampled-token scores. Exact forward or reverse KL and total variation aggregate over vocabulary distributions; the sampled-token discrepancy matches the information stored in the replay buffer.

Let K be the number of retained rollout blocks, B the number of prompt groups per block, n the number of responses per group, and L the average number of valid response tokens. The buffer contains $Q = KB$ groups, $N = Qn$ responses, and approximately NL valid response tokens. We denote the current-actor forward cost for one candidate response by $C_{\mathrm{fwd}}(\theta, L)$, including its prompt context and the model configuration.

D-ARL evaluates the current actor on all N candidates before constructing its replay scores. GA-ARL instead performs elementwise operations on cached behavior/reference log-probabilities and advantages, followed by masked response means and group means. Its token scoring and aggregation cost is $O(NL)$. The shared historical scaling statistic is computed from scalar records; averaging the H records collected so far costs $O(H)$ in the implementation.

Both methods use the same group-selection operation. In the code, this is a top-M call with `largest=False` and `sorted=False`, followed by gathering all n responses for each selected group. Let $C_{\mathrm{top}}(Q, M)$ denote that operation's cost. The replay-selection costs are

$$\begin{aligned} C_{\text{D-ARL}} &= O(NC_{\mathrm{fwd}}(\theta, L) + NL + H) + C_{\mathrm{top}}(Q, M), \\ C_{\text{GA-ARL}} &= O(NL + H) + C_{\mathrm{top}}(Q, M). \end{aligned} \tag{49}$$

The top-M operation selects groups without requesting a fully sorted output. Response gathering copies the selected sequences in both methods. Table 14 summarizes the main operations.

Table 14: Replay-selection operations. $N = KBn$ is the number of candidate responses and $Q = KB$ the number of prompt groups. H is the number of historical scalar records.

Operation	D-ARL	GA-ARL
Candidate current-actor evaluation	$O(NC_{\mathrm{fwd}})$	—
Token scoring and aggregation	$O(NL)$	$O(NL)$
Historical scalar average	$O(H)$	$O(H)$
Group selection	$C_{\mathrm{top}}(Q, M)$	$C_{\mathrm{top}}(Q, M)$
Selected-batch actor optimization	Shared training procedure	Shared training procedure

Token masks, cached log-probabilities, and scoring tensors require $O(NL)$ storage; group scores require $O(Q)$ storage, and the historical scalar list uses $O(H)$. Reference log-probabilities are computed when a rollout block enters the buffer in both replay variants and reused during selection. Actor forward and backward passes on the selected batch remain part of the common training pipeline. GA-ARL's computational saving comes from eliminating the candidate-wide current-actor evaluation, whose cost grows with the number of replay candidates, sequence lengths, and model size. End-to-end runtime also reflects rollout overlap, batching, padding, data movement, and validation.

F.4 Algorithm of GA-ARL

Algorithm 1 summarizes GA-ARL. Each retained block contains prompt groups, stored rewards, response masks, and cached behavior/reference log-probabilities. Scoring uses the historical scalar in Appendix F.2, and selection preserves complete GRPO groups.

Algorithm 1 Gradient-Aware Asynchronous Reinforcement Learning

Initialize actor π_θ, reference π_{ref}, and replay buffer $\mathcal{B}$
Set block limit K, warmup step T_{warm}, and selected group count M
Set $\beta_{\mathrm{target}} = 1$, $\lambda_{\mathrm{KL}} = 0.001$, and $\epsilon = 10^{-8}$
Initialize historical statistics $\mathcal{H} \leftarrow \emptyset$; start rollout workers
for training step $t = 1, 2, \ldots$ **do**
    Receive a completed rollout block $\mathcal{D}_{\mathrm{new}}$ with behavior version k
    Assign prompt-group IDs, evaluate rewards, and cache ℓ_i^{b}, ℓ_i^{ref}, m_i
    Insert $\mathcal{D}_{\mathrm{new}}$ into $\mathcal{B}$; retain the most recent K blocks
    if $t < T_{\mathrm{warm}}$ **then**
        $\mathcal{D}_{\mathrm{sel}} \leftarrow \mathcal{D}_{\mathrm{new}}$
    else
        Compute group-relative advantages from the rewards in $\mathcal{B}$
        $\widehat{z}_t \leftarrow \mathrm{mean}(\mathcal{H})$
        for each response $\tau \in \mathcal{B}$ **do**
            for each valid response token i **do**
                $v_i \leftarrow \ell_i^{\mathrm{ref}} + A_i + \log|(A_i + \epsilon)/(\widehat{z}_t + \epsilon)|$
                $d_i \leftarrow 1 - \exp(-|v_i - \ell_i^{\mathrm{b}}|)$
            end for
            Compute masked response score using Eq. (11)
        end for
        Average response scores within each prompt group
        Select M lowest-scoring groups using Eq. (12)
        $\mathcal{D}_{\mathrm{sel}} \leftarrow$ all responses in the selected groups
    end if
    Launch the next rollout asynchronously with the latest available actor weights
    Recompute GRPO advantages for $\mathcal{D}_{\mathrm{sel}}$
    Update the actor on $\mathcal{D}_{\mathrm{sel}}$ using Eq. (13)
    Compute masked mean advantage $\bar{A}_t$ on $\mathcal{D}_{\mathrm{sel}}$
    Append $\bar{A}_t \exp(\bar{A}_t)$ to $\mathcal{H}$
end for

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe replay selection for asynchronous LLM post-training and its accuracy–efficiency benefits. The experiments evaluate Qwen3-1.7B and Qwen3-4B on five reasoning benchmarks, with multi-seed accuracy and timing results in Appendices D.3 and D.4, and the long-response evaluation in Appendix D.7.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A discusses advantage-estimation quality, sparse and systematically biased rewards, the local-update and comparable-score-function-norm assumptions, and the range of models and tasks evaluated. Reward-noise sensitivity is examined in Appendix D.6, and replay-selection scaling is analyzed in Appendix F.3.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theorem 4.1 studies the fixed-state KL-regularized objective. Proposition 4.2 states the local-update and comparable-score-function-norm assumptions. Their derivations are given in Appendices B.1 and B.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The public implementation and configuration guide are available at `https://github.com/MIRALab-USTC/GA-ARL`. Dataset sources and training settings are described in Appendices E and F.1. Appendices F.2 and F.4 specify replay scoring and the algorithm; Appendices D.3, D.4, and D.6 provide seed-level results and statistical procedures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, installation instructions, configuration documentation, and experiment scripts are public at `https://github.com/MIRALab-USTC/GA-ARL`. The repository documents environment setup, model and data paths, and training commands. Public dataset sources, training configurations, and evaluation protocols for the main experiments are provided in Appendices E and F.1.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The main experimental section reports tasks, model backbones, baselines, and metrics. Data splits and training hyperparameters are provided in Appendix E and Table 13. The additional multi-seed, synchronous GRPO, reward-noise, and long-response comparisons are described in Appendices D.3, D.4, D.5, D.6, and D.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Eight matched seeds (42–49) are reported for Qwen3-4B GSM8K, Qwen3-4B LiveCodeBench, Qwen3-1.7B HumanEval, and the reward-noise comparisons. Tables report means and sample standard deviations across seeds. Appendices D.3, D.4, and D.6 give two-sided paired t-tests and 95% confidence intervals for paired accuracy differences; the two reward-noise degradation tests use Holm correction.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F.1 reports four NVIDIA A800-SXM4 GPUs with 80GB memory each for all asynchronous experiments, with two GPUs for rollout generation and two for policy training. Experiment 2 and Appendix D.2 report wall-clock training curves; Appendices D.3 and D.4 provide multi-seed training-time reductions. Appendix D.7 specifies the common 66-hour budget for the long-response comparison, corresponding to 264 GPU-hours per method. Appendix F.3 analyzes replay-selection and storage costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The research focuses on improving the efficiency and effectiveness of asynchronous reinforcement learning for LLM post-training, and the experiments use standard public benchmarks and do not involve human-subject studies, private user data, or sensitive personal information. We have reviewed the NeurIPS Code of Ethics and believe the work conforms to it.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses broader impacts in Appendix A, including positive impacts such as reduced RL post-training cost and improved accessibility, as well as potential risks such as more efficient optimization of harmful models or amplification of biased reward signals.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk model, image generator, or scraped dataset; experiments use existing public benchmarks and datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources for all existing datasets, benchmarks, models, and training frameworks used in the experiments. Appendix E.5 identifies the released datasets and software used and summarizes their declared licenses. Ordering Puzzle and Zebra Puzzle use the official Guru evaluation release; its dataset card declares the MIT License. The original benchmark works are cited separately.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released GA-ARL code at `https://github.com/MIRALab-USTC/GA-ARL` includes installation instructions, an experiment configuration guide, and a script index documenting the available training and validation entries. The repository retains the upstream VERL copyright notices and Apache 2.0 license, and describes the coverage of the released experiment scripts.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments, user studies, or research with human subjects. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use LLMs as an important, original, or non-standard component beyond standard LLM policy training and evaluation. Any use of LLM-based tools for writing, editing, formatting, or basic code assistance does not affect the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.