AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning

Wei Fu[12]*, **Jiaxuan Gao**[1], **Xujie Shen**[2], **Chen Zhu**[2], **Zhiyu Mei**[12], **Chuyi He**[2], **Shusheng Xu**[12], **Guo Wei**[2], **Jun Mei**[2], **Jiashu Wang**[3], **Tongkai Yang**[2], **Binhang Yuan**[3], **Yi Wu**[1]

[1] IIIS, Tsinghua University, [2] Ant Group, [3] HKUST
fuwth17@gmail.com, jxwuyi@gmail.com

Abstract

Reinforcement learning (RL) has become a trending paradigm for training large language models (LLMs), particularly for reasoning tasks. Effective RL for LLMs requires massive parallelization and poses an urgent need for efficient training systems. Most existing large-scale RL systems for LLMs are synchronous by alternating generation and training in a batch setting, where the rollouts in each training batch are generated by the same (or latest) model. This stabilizes RL training but suffers from severe system-level inefficiency. Generation must wait until the longest output in the batch is completed before model update, resulting in GPU underutilization. We present AReaL, a *fully asynchronous* RL system that completely decouples generation from training. Rollout workers in AReaL continuously generate new outputs without waiting, while training workers update the model whenever a batch of data is collected. AReaL also incorporates a collection of system-level optimizations, leading to substantially higher GPU utilization. To stabilize RL training, AReaL balances the workload of rollout and training workers to control data staleness, and adopts a staleness-enhanced PPO variant to better handle outdated training samples. Extensive experiments on math and code reasoning benchmarks show that AReaL achieves **up to 2.77× training speedup** compared to synchronous systems with the same number of GPUs and matched or even improved final performance. The code of AReaL is available at https://github.com/inclusionAI/AReaL/.

1 Introduction

Reinforcement learning (RL) has emerged as a new scaling paradigm for enhancing the capabilities of large language models (LLMs) by enabling thinking abilities [52]. Given a prompt, RL allows an LLM to generate thinking tokens before outputting a final answer, enabling *test-time scaling* [29, 47]. These thinking LLMs are named Large Reasoning Models (LRMs) and have been shown to have particularly strong capabilities on challenging reasoning problems, such as math [9, 5, 20], coding [3, 14, 15], logic puzzles [22, 34], and agentic tasks [23, 57].

Effective RL training often requires massive parallelization to derive a large batch of rollouts for sufficient exploration, which is the key to obtaining optimal model performance. For example, popular RL algorithms, such as PPO [42] and GRPO [43], often require an effective training batch of thousands of outputs [60, 61, 53]. Moreover, an LRM can generate tens of thousands of thinking tokens for each input prompt [6], further posing an urgent need for an efficient training system to run RL training on a large scale.

*This work was conducted during an internship at Ant Group.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

However, developing an efficient large-scale RL system is challenging. An RL system needs to frequently switch between LLM generation and training, which can introduce significant system overhead without careful optimizations. For LRMs, the output length of the training model varies significantly for different prompts throughout the RL process, which results in an ever-changing workload for both generation and training. This characteristic often triggers idle time in high-performance hardware, leading to a waste of computation. Furthermore, classical large-scale RL algorithms like PPO or GRPO typically require on-policy training data, i.e., samples generated by the latest model, to ensure the best model performance, which poses additional system challenges.

Consequently, most existing large-scale RL systems are designed in a fully synchronous manner [27, 11, 45, 44] by strictly alternating between LLM generation and training, which ensures that the LLM is always trained on the latest outputs for the best practical performance. In such a synchronous design, the generation step must wait until the finish of the longest output within a batch. Due to the varying output lengths for LRMs, a synchronous RL system suffers from severe training inefficiency. Very recently, there have also been attempts to explore parallel generation and training [30, 24, 49]. These works use outputs generated from a previous model version to update the current model. For the best performance, the model version used for rollout generation is limited to only one or two steps older. However, all these systems still follow a batched generation setting, where all the samples within a training batch are from the same model version. Accordingly, the issue of system inefficiency during the generation phase still remains unaddressed.

To fundamentally resolve the issues in system design, we develop AReaL, a fully Asynchronous RL training system for LRMs that completely decouples generation from training without hurting the final performance. AReaL runs LLM generation in a streaming manner, where each rollout worker continuously generates new outputs without waiting, leading to high GPU utilization. Meanwhile, the trainer workers in AReaL run parallel model updates whenever a training batch is obtained from the rollout workers. Once the model is updated, we synchronize the model weights in each rollout worker. In such an asynchronous design, each training batch of AReaL may contain samples generated by different model versions. Therefore, AReaL incorporates a modified objective of the PPO algorithm, which can leverage samples generated from much older model versions without any performance drop. AReaL also conducts a data filtering process to ensure the staleness of each training sample is well controlled. In addition, AReaL also introduces several system-level optimizations, including interruptible rollout workers, dynamic batching for variable-length outputs, and parallel reward service, which further improve the overall training throughput.

We evaluate AReaL on challenging mathematical reasoning and code generation tasks using models up to 32B parameters. Compared to state-of-the-art synchronous systems, AReaL achieves up to $2.57\times$ higher training throughput and linear scaling efficiency up to 512 GPUs. Crucially, this acceleration *even comes with improved solution accuracy on these tasks*, illustrating that AReaL delivers significant efficiency gains without sacrificing (and indeed enhancing) model performance.

sectionRelated Work

RL for LLMs Reinforcement learning (RL) has emerged as the predominant paradigm for enhancing the reasoning capabilities of Large Language Models (LLMs) [31, 32]. Existing RL approaches typically focus on tasks with well-defined reward functions, including mathematical reasoning [9], coding [14, 15], scientific problem solving [39, 36], and tool use [57]. During training, models learn to reason by progressively extending the length of chain-of-thought trajectories [52, 6]. Recent open-source initiatives have demonstrated significant success in improving model capabilities through smaller distilled models [24, 25]. Our work builds upon this research direction, distinguishing itself from preference-based RLHF [33] and zero-shot reasoning approaches [60, 61, 12] that attempt to acquire reasoning skills from pre-trained models without task-specific fine-tuning.

Asynchronous RL The decoupled asynchronous RL architecture [21, 8, 26], combined with corresponding algorithmic innovations [7, 16], has achieved remarkable success in game applications [2, 51]. Although similar asynchronous approaches have been explored for LLM training, they typically focus on short-context settings [30, 1, 40] (e.g., RLHF) or one/two-step generation-training overlap [24, 48]. Our work extends these studies and provides a more flexible trade-off between staleness and training speed, as we will show in Section 4. In contrast to concurrent work [64] that maximizes system-level efficiency, we adopt an algorithm-system co-design approach that provides both an expressive system and a practical algorithm implementation. Our interruptible generation technique is conceptually similar to partial rollout [17] in synchronous RL systems. Instead of

setting a fixed length budget, AReaL dynamically interrupts generation while maintaining consistent training batch sizes through buffering, thus preserving the stability of PPO. Compared with prior methods [40, 30], our algorithmic innovation in the asynchronous setting can endure higher data staleness and remains compatible with interruptible generation.

LLM Training and Inference Our work focuses on dense transformer models [50]. The RL training primarily consists of generation (inference) and training phases. Generation involves auto-regressive decoding, which requires efficient KV cache management [63, 18] and optimized decoding kernels [58]. Training requires careful orchestration of data, tensor, and pipeline parallelism strategies [38, 46, 62]. While conventional synchronous systems execute generation and training sequentially on the same hardware resources, they require different optimal parallelization strategies. Recent work has proposed context switching [19, 17] or weight resharding [45, 27] techniques to address this mismatch. AReaL advances beyond synchronous RL systems by decoupling generation and training, completely eliminating resharding overhead from the critical training path.

2 Background

2.1 Preliminaries about RL Training

RL Formulation and PPO We formulate our problem within the Markov Decision Process (MDP) framework [37], defined by the tuple $\langle \mathcal{S}, \mathcal{A}, r, P, \gamma, H \rangle$. Here, $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, P the transition model, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, γ the discount factor, and H the horizon. The LRM implements a parameterized policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ where each action $a_t \in \mathcal{A}$ corresponds to a text token from the vocabulary. The state $s_t \in \mathcal{S}$ consists of a question $s_1 = q$ followed by previously generated response tokens $(a_1, .., a_{t-1})$, with deterministic transitions $s_{t+1} = \text{concat}(s_t, a_t)$. Given a question distribution $\mathcal{D}$, we optimize the objective:

$$J(\theta) = \mathbb{E}_{q \sim \mathcal{D}, a_t \sim \pi_\theta(\cdot|q, a_{<t})} \left[\sum_{t=1}^{H} \gamma^{t-1} r(s_t, a_t) \right]. \tag{1}$$

Following common practice [6, 25], we use a rule-based reward function that only provides non-zero feedback on the final action, indicating answer correctness, and set $\gamma = 1$. We optimize this objective using Proximal Policy Optimization (PPO) [42]:

$$J_{\text{PPO}}(\theta) = \mathbb{E}_{q \sim \mathcal{D}, a_t \sim \pi_{\text{old}}(\cdot|q, a_{<t})} \left[\sum_{t=1}^{H} \min \left(u_t(\theta) \hat{A}(s_t, a_t), \text{clip}\left(u_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}(s_t, a_t) \right) \right], \tag{2}$$

where $u_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ denotes the importance ratio and $\hat{A}(s_t, a_t)$ represents the estimated advantage [41]. Following standard practices in RL [42, 33], we divide the global batch into minibatches for sequential parameter updates.[2]

Distributed Systems for LRM Training Our work focuses on enhancing reasoning capabilities for LRMs after Supervised Fine-Tuning (SFT), distinct from approaches that incentivize reasoning in pre-trained base models [6]. LRMs after SFT produce long reasoning sequences (e.g., 32K tokens) and usually require large global batch sizes (e.g., 128 prompts with 16 responses each) for stable RL training [6, 25, 24, 60, 61]. In *synchronous RL systems*, two phases are iteratively executed: generation (rollout) and training. The generation phase uses the latest model parameters to produce multiple reasoning traces for each query in the training batch. The training phase then updates the model parameters based on the generated trajectories. These phases execute iteratively on the same GPUs.

2.2 Motivation for Asynchronous RL System

We identify two essential limitations in synchronous RL systems:

Inference devices are underutilized. As shown in Figure 1 (left), generation must wait for the longest sequence to complete before training can begin. This leads to non-uniform decoding length across GPUs, which underutilizes GPU compute resources.

[2]This differs from gradient accumulation, which performs a single update across minibatches.

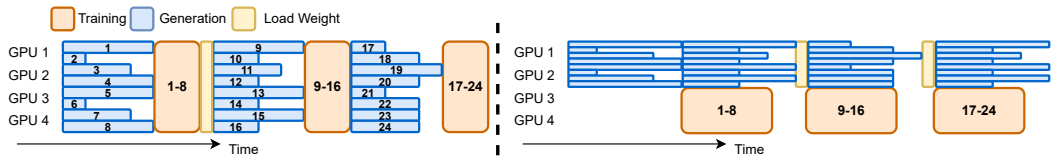


Figure 1: Execution timeline of a synchronous (left) and a one-step overlap (right) RL system showing underutilized inference devices.

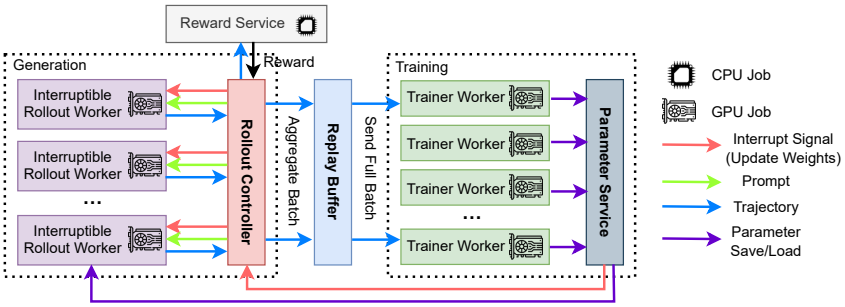


Figure 2: The AReaL architecture featuring asynchronous generation and training components.

Scalability is poor in synchronous RL systems. Synchronous systems distribute generation across all devices, reducing the per-GPU decoding batch size. This pushes the decoding process into a memory-IO-bound regime [4, 28] where additional devices fail to improve throughput.

3 System Architecture

The limitations identified in Section 2.2 motivate our design of a system that fully decouples generation and training across separate GPU clusters. This system should be hardware-efficient, scalable, and equipped with the flexibility for a customized RL workflow. We implement these principles in AReaL, an asynchronous RL system specifically designed for efficient large-scale LRM training.

3.1 System Overview

Figure 2 presents the architecture and data flow of AReaL. The system comprises 4 core components:

Interruptible Rollout Worker handles two types of requests: (1) The `generate` request generates responses given prompts. (2) The `update_weights` request interrupts all ongoing generations and loads parameters of new versions. Upon the interruption, the rollout workers discard KV caches computed by old weights, and re-compute them using the new weights. Afterwards, the rollout workers continue to decode the unfinished sequences until the next interruption or termination. We emphasize that such interruptions and in-flight weight updates would result in trajectories composed of segments produced by different model versions. This introduces a novel algorithmic challenge, which will be addressed in Section 4.

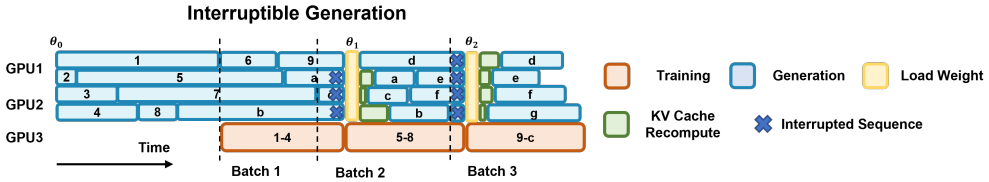


Figure 3: Illustration of generation management in AReaL. Vertical lines show the ready time for the next step training. Blue crosses show the interrupted requests when new parameters arrive.

Reward Service evaluates the accuracy of the responses generated by the model. For example, in the coding task, this service extracts the code and executes unit tests to verify its accuracy.

Trainer Workers continuously sample from the replay buffer, accumulating data until reaching the configured training batch size. They then perform PPO updates and store the resulting parameters in distributed storage. To ensure data freshness, data from the replay buffer is used only once.

Rollout Controller serves as a critical bridge between the rollout workers, reward service, and the model workers. During the training process, it reads data from the dataset and invokes the rollout worker's `generate` request. The received response is then sent to the reward service to obtain the reward. The trajectory, along with the reward, is stored in the replay buffer, waiting to be trained by the model worker. After the model worker updates the parameters, the controller calls the rollout worker's `update_weight`. We illustrate the generation and training management in Figure 3. This asynchronous pipeline ensures continuous full utilization of both generation and training resources.

3.2 Algorithmic Challenges

While the asynchronous system design offers significant acceleration through improved device utilization, it introduces several technical challenges that require algorithmic considerations.

Data Staleness Due to the asynchronous nature of AReaL, each training batch contains data from multiple prior policy versions. Prior works on asynchronous RL training systems have demonstrated that such staleness can degrade learning performance in both RLHF [30] and game environments [2]. Data staleness leads to a distribution gap between the training data and the latest model. In asynchronous RL training for LRMs, this issue could be even more severe for long trajectories due to extended decoding time.

Inconsistent Policy Versions As discussed in Section 3.1, the generated trajectories may involve segments produced by different policy versions. This inconsistency fundamentally violates the formulation of standard PPO in Eq. 2 that assumes all actions are generated by a single policy π_{old}.

In the following section, we detail our technical innovations for overcoming these challenges while preserving the efficiency advantages of an asynchronous system.

4 Addressing the Algorithmic Challenges in AReaL

4.1 Staleness-Aware Training

To avoid the performance drop due to training on data with extremely high staleness, we introduce a hyperparameter η representing *the maximum permitted staleness in each training batch* for staleness-aware training. In particular, when $\eta = 0$, our system degenerates to synchronous RL with all training samples generated by the current policy. We implement the staleness control in our system by dynamically controlling the throughput of the generation requests sent to the generation servers. Given the current policy version i, the total number of generated trajectories N_r, and the training batch size B for each training step, we enforce the following formula whenever submitting new generation requests:

$$\lfloor (N_r - 1)/B \rfloor \leq i + \eta. \tag{3}$$

We also prioritize older trajectories from the data buffer to form a training batch. In our system implementation, the rollout controller tracks both the generated samples N_r and policy version i from the parameter server. It rejects new generation requests that may violate the staleness constraint.

Note that this rate-limiting protocol is a simple yet effective design choice in practice. However, when η is too small, the generation throughput can be slowed down when some extremely long trajectories are being generated. Therefore, we empirically suggest adopting a large staleness-control parameter η for the best system throughput. This system-wide practice also motivates us to apply an enhanced algorithm that can make effective use of more stale data for RL training.

4.2 Decoupled PPO Objective

We apply a decoupled PPO objective [10] that disentangles the *behavior policy* and the *proximal policy*. The behavior policy π_{behav} represents the policy used for sampling trajectories and the

proximal policy π_{prox} is a proximal policy serving as a recent target to regularize the update of π_θ. By applying importance sampling on the sampled trajectories, we derive a decoupled PPO objective suitable for asynchronous RL training:

$$J(\theta) = \mathbb{E}_{q\sim\mathcal{D},a_t\sim\pi_{\text{behav}}}\left[\sum_{t=1}^{H}\min\Big(\underbrace{\boxed{\frac{\pi_\theta}{\pi_{\text{behav}}}}}_{\text{Importance Ratio}}\hat{A}_t,\ \overbrace{\frac{\pi_{\text{prox}}}{\pi_{\text{behav}}}\text{clip}\Big(\underbrace{\boxed{\frac{\pi_\theta}{\pi_{\text{prox}}}}}_{\text{Trust Region Center}},1-\epsilon,1+\epsilon\Big)\hat{A}_t}^{\text{Importance Ratio}}\Big)\right] \tag{4}$$

$$= \mathbb{E}_{q\sim\mathcal{D},a_t\sim\pi_{\text{behav}}}\left[\sum_{t=1}^{H}\frac{\pi_{\text{prox}}}{\pi_{\text{behav}}}\min\left(u_t^{\text{prox}}(\theta)\hat{A}_t,\text{clip}\left(u_t^{\text{prox}}(\theta),1-\epsilon,1+\epsilon\right)\hat{A}_t\right)\right], \tag{5}$$

where $u_t^{\text{prox}}(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{prox}}(a_t|s_t)}$ is the importance ratio with respect to the proximal policy. We omit the state-action terms for conciseness.

The main difference between the asynchronous PPO objective in Equation 5 and the standard one in Equation 2 lies in the proximal policy π_{prox} for regularizing the model update. In asynchronous PPO training, using the behavior policy as the proximal policy will pull the latest policy π_θ towards the old-version and low-quality policies, thus slowing down model improvements. By employing a recent policy as the proximal policy, model updates happen within the trust region around the high-quality proximal policy π_{prox}, thus stabilizing training.

The decoupled PPO objective in Equation 5 provides a natural benefit: it relaxes the requirement that all data within one training batch should be generated with a single policy. This property is crucial for maintaining algorithmic correctness when combining interruptible generation with policy updates. We claim that the inconsistent policy versions across a trajectory maintain equivalence to a single behavior policy π_{behav}. (See Section D for the proof.)

Proposition 1. *For any sequence $(q, a_1, \ldots, a_H)$ generated by policies $(\pi_\theta, \ldots, \pi_{\theta+k})$ where $\pi_{\theta+i}$ produces tokens $(a_{t_i}, \ldots, a_{t_{i+1}})$, where $1 = t_0 < \cdots < t_{k+1} = H$, there exists a behavior policy π_{behav} such that the interrupted generation is equivalent to sampling entirely from π_{behav}.*

Practical Remark While Hilton et al. [10] maintains an exponential moving average of parameters for π_{prox}, this approach is prohibitively expensive for LRMs. Consequently, we simply use the parameters before each model update step as π_{prox}. Equation 5 is implemented by recomputing token probabilities upon the arrival of the global batch in each training step.

5 Implementation

We implement AReaL using Python and PyTorch [35] built upon the ReaLHF [27] framework. Our system combines SGLang [63] v0.4.6 for generation serving with Megatron-Core [46] v0.11.0 as the training backend, managed by SLURM [59] for resource scheduling. To maximize throughput for both generation and training phases, we implement several key system-level optimizations that address critical bottlenecks in the pipeline.

AReaL decouples GPU computation from CPU operations, including rule-based reward computation (such as string matching for math problems or unit test execution for code) and TCP-based data transfer. By executing these operations in separate threads and pipelining the workflow, we overlap reward computation and data transfer with subsequent generation requests. We use asyncio coroutines to concurrently run multiple requests in the rollout worker to avoid mutual blocking waits.

To handle training with variable-length sequences, we employ a padding-free sequence packing strategy coupled with a dynamic allocation algorithm. The algorithm balances token distribution across micro-batches under fixed memory constraints (see Algorithm 1). This approach maximizes GPU memory utilization while minimizing the number of required forward-backward passes.

6 Experiments

Our evaluation comprises three components: (1) comprehensive comparisons against state-of-the-art open-source frameworks across model sizes, (2) strong-scaling analysis with varying compute resources, and (3) ablation studies validating our design choices.

6.1 Experiment Setup

We evaluate AReaL on challenging math and coding tasks. We employ the distilled Qwen2 model series [54, 55] from DeepSeek-R1 [6] as base models (i.e., R1-Distilled-Qwen), spanning from 1.5B to 32B parameters. For each task-model combination, we train for a fixed number of PPO updates and evaluate the final checkpoint. Our evaluation of mathematical tasks follows the Qwen evaluation protocol [56, 13], while coding models are assessed on LiveCodeBench (8/1/24-2/1/25) [14] using the official protocol. Unless otherwise specified, we set the maximum staleness $\eta = 4$ for coding and $\eta = 8$ for math, and adopt the training configurations used in Section 6.2, with additional hyperparameters detailed in Appendix B.

We conduct experiments on an H800 GPU cluster comprising 64 nodes, each equipped with 8 GPUs. The cluster features NVLink for intra-node connectivity and RoCE with 3.2Tbps bandwidth for inter-node communication. To ensure rapid convergence, we allocate a minimum of 16 nodes as a baseline pod configuration for complete experiments. We scale the number of nodes proportionally with model size, ultimately utilizing 48 nodes for training our largest 32B parameter model. This scaling strategy enables us to run experiments of varying sizes in parallel while maintaining efficient resource utilization.

For AReaL, we maintain a fixed ratio between inference and training devices, allocating three-quarters of the devices for inference. This configuration was selected over an equal 50-50 partition based on our early experiments, where the 75-25 partition demonstrated higher training throughput. Although we adopt this ratio as a heuristic configuration, we emphasize that the optimal partition may vary across different settings and could potentially benefit from dynamic adjustment during training, as discussed in Section 7.

6.2 End-to-End Comparison

We establish two state-of-the-art baselines using synchronous RL systems: DeepScaleR [25] for mathematical reasoning with a 1.5B model, and DeepCoder [24] for code generation with a 14B model, both trained using verl [45]. For larger 7B and 32B models where comparable baselines are unavailable, we performed controlled experiments by training from scratch using a synchronous variant of AReaL. After training, we evaluate on the challenging AIME24 benchmark for math models and the LiveCodeBench [14] benchmark for coding models. Evaluation results on additional benchmarks are presented in Appendix C.

Our main results are shown in Table 1. Since the code for obtaining previous SOTA models can be out-of-date, we measure the throughput and estimate the training hours using the latest verl code for a fair comparison. AReaL consistently matches or exceeds baseline performance while achieving significant speedups without performance degradation. In particular, the end-to-end training time can be reduced by $2.77\times$ compared with synchronous systems.

6.3 Scalability

We compare the scalability of AReaL with verl [45], the state-of-the-art synchronous RL system, across different model sizes and context lengths. We select the minimum number of GPUs when verl does not encounter the OOM issue for 7B models and 32k context length, then we proportionally adjust the number of GPUs according to the model size. We measure the *effective throughput* for training, defined as the rate of consuming generated tokens during PPO updates, after proper warmup steps. Figure 4 presents the results for context lengths of 16k and 32k. Here, context length refers to the sum of prompt length and generated length, with the maximum prompt length capped at 1k.

Across all settings, AReaL demonstrates an approximate linear scaling trend with increased device count, while the synchronous system typically fails to scale effectively. AReaL's throughput surpasses the baseline in most settings, and could achieve at most $2.5\times$ speedup. We note that for

Table 1: End-to-End Performance Comparison. We evaluate on the AIME24 benchmark for math and LiveCodeBench (8/1/24-2/1/25) for coding. We limit the maximum generation length to 32K tokens and sample 32 responses per question, reporting the average pass@1 accuracy. * represents the best known reproducible results obtained via RL, as cited from DeepScaler [25] and DeepCoder [24] respectively. AReaL achieves comparable performance with 2× fewer training hours.

Model	AIME24 ↑	# Nodes	PPO Steps	Training Hours ↓
1.5B basemodel	29.3	-	-	-
w/ VeRL	**43.1***	16	250	33.6
w/ Sync.AReaL	42.0	16	250	41.0
w/ AReaL (ours)	42.2	16	250	**14.8**
7B basemodel	54.3	-	-	-
w/ VeRL	-	24	250	52.1
w/ Sync.AReaL	63.0	24	250	57.7
w/ AReaL (ours)	**63.1**	24	250	**25.4**

Model	LiveCodeBench ↑	# Nodes	PPO Steps	Training Hours ↓
14B basemodel	53.4	-	-	-
w/ VeRL	57.9*	32	80	44.4
w/ Sync.AReaL	56.7	32	80	48.8
w/ AReaL (ours)	**58.1**	32	80	**21.9**
32B basemodel	57.4	-	-	-
w/ VeRL	-	48	60	46.4
w/ Sync.AReaL	**61.2**	48	60	51.1
w/ AReaL (ours)	61.0	48	60	**31.1**

smaller context lengths, the advantage of AReaL can be smaller because the generation throughput cannot match the pace of training throughput. Although many sequences are generated, they are not effectively consumed by the training process. Additionally, AReaL is more robust with longer generation lengths due to asynchronous and interruptible generation. The generation of long responses can be fully hidden in the critical path, so extending generation length does not drastically affect the effective training throughput of AReaL.

6.4 Algorithm Ablations

We conduct ablation studies to validate our algorithmic innovations in Section 4 by training a 1.5B LRM on math tasks. We follow the basic experiment setting of DeepScaleR and then gradually

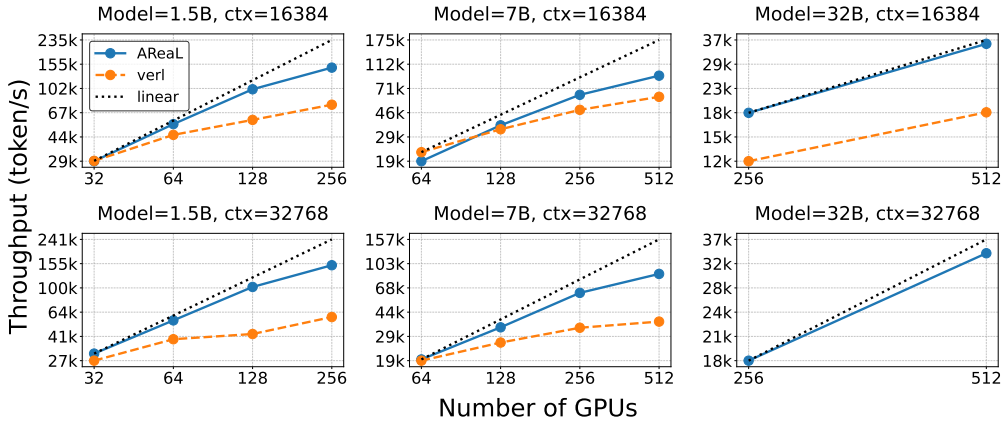
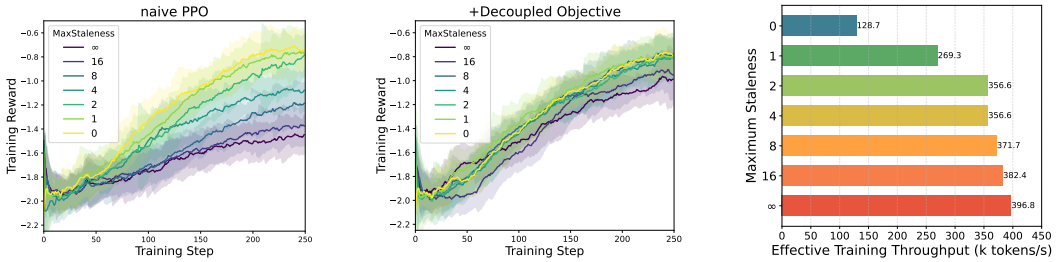


Figure 4: The strong scaling trend. Dotted lines indicate ideal linear scaling. verl consistently encounters OOM with 32k context length and the 32B model so the data points are missing.



(a) Learning curves with naive PPO.

(b) Learning curves with eq. (5).

(c) Effective training throughput.

Figure 5: Ablation studies of the decoupled PPO objective and staleness control with a 1.5B model on math reasoning tasks. Both algorithmic choices are essential. With a moderate staleness value and the decoupled objective, training progress can be accelerated by over $2\times$ while maintaining final evaluation performance.

Table 2: Evaluation scores when varying data staleness, comparing performance with and without the decoupled objective. Numbers within ± 1 of the oracle score are underlined.

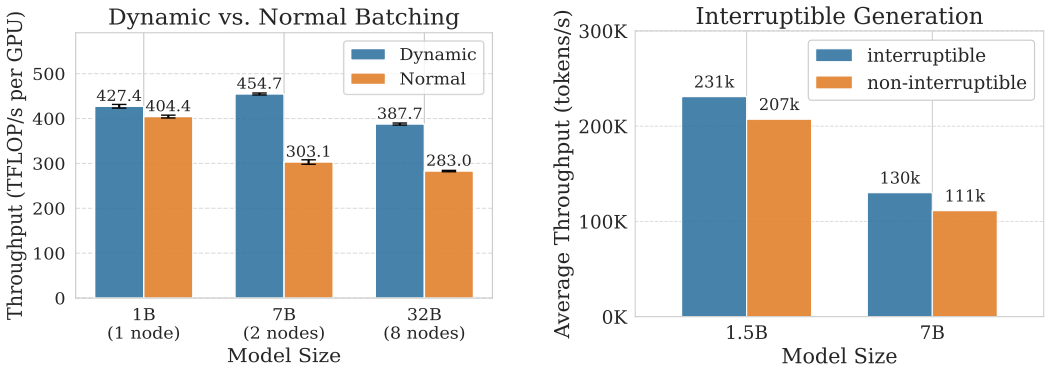
Max.Stale.	AIME24		AIME25		AMC23		MATH 500	
	W/o	With	W/o	With	W/o	With	W/o	With
0 (Oracle)	42.0		32.9		84.4		89.2	
1	41.8	42.1	30.7	31.9	83.3	85.2	89.9	89.8
2	40.0	41.8	32.1	32.5	82.3	84.3	89.6	89.6
4	23.3	42.2	23.1	32.0	58.5	85.1	66.9	89.5
8	35.7	41.0	27.8	31.1	81.2	82.9	87.8	89.2
16	35.8	38.7	26.2	32.5	78.4	83.2	87.4	89.1
∞	34.0	36.9	26.9	29.9	79.4	81.0	87.1	88.1

increase the η value for ablation purposes. Specifically, we vary the maximum allowed staleness η and compare configurations with and without the decoupled PPO objective. Figures 5a and 5b show the learning curves after 250 training steps. Table 2 presents the corresponding final evaluation performances across multiple mathematical reasoning benchmarks. We follow the common practice of PPO and perform multiple mini-batch updates within each training step. We emphasize that η constrains the training batch staleness regarding training steps.

Figure 5a demonstrates that naive PPO fails to match the performance of the synchronous RL oracle (i.e., the performance when $\eta = 0$). Even slight staleness can significantly degrade final performance due to the improper clipping center and policy changes during interruptible generation. Furthermore, increasing data staleness consistently degrades learning performance, aligning with observations from prior work in other domains [2, 30]. However, as shown by comparing Figure 5b and Figure 5a, the decoupled PPO objective substantially improves training stability when handling stale data, consistent with findings from [10] in game domains. In addition, we observe that even with the decoupled objective, unbounded staleness (maximum staleness $\to \infty$) still results in inferior performance compared to the zero-staleness oracle. When properly constrained, moderate staleness (e.g., $\eta \leq 8$) has minimal impact on final performance while significantly accelerating training through the asynchronous pipeline, as demonstrated in Figure 5c and Table 2. These results validate our approach of combining controlled staleness with the decoupled PPO objective for efficient asynchronous RL training.

6.5 System Ablations

Dynamic Microbatch Allocation We investigate the effectiveness of dynamic batching by comparing PPO training throughput against a standard micro-batching strategy. The standard micro-batching strategy can result in multiple long sequences being assigned to the same micro-batch, thus usually requiring a sufficiently large number of micro-batches to prevent out-of-memory errors. In our



(a) Ablation study of dynamic micro-batch allocation.

(b) Ablation study of interruptible generation.

Figure 6: Ablation studies on system optimizations.

experimental setup, we configured 32 micro-batches for the standard setting and established a token budget of 32,768 per micro-batch for the dynamic batching approach. As demonstrated in Figure 6a, dynamic batching yields an average of 30% throughput improvements across various model sizes.

Interruptible Generation We ablate interruptible generation and present the resulting generation throughput in Figure 6b. Without interruptible generation, the controller must wait for the longest response. In particular, interruptible generation leads to a 12% and 17% throughput increase for 1.5B and 7B models respectively on 4 nodes, which validates our architectural design choice.

7 Conclusion

This paper introduces AReaL, a fully asynchronous system designed for efficient large-scale reinforcement learning (RL) training. The AReaL architecture provides both the flexibility and expressiveness required for implementing asynchronous algorithms. Building upon this foundation, we contribute several algorithmic innovations, including staleness-aware training and a decoupled PPO objective, which enable efficient and stable PPO training in asynchronous environments. Our experimental results demonstrate AReaL's superior hardware efficiency, sample efficiency, and scalability compared to existing synchronous RL systems. This work provides a starting point for reliably scaling RL training. We hope that it can enable future advances in large-scale AI systems that push the boundaries of machine intelligence further.

References

[1] B. R. Bartoldson, S. Venkatraman, J. Diffenderfer, M. Jain, T. Ben-Nun, S. Lee, M. Kim, J. Obando-Ceron, Y. Bengio, and B. Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable LLM post-training. *CoRR*, abs/2503.18929, 2025. doi: 10.48550/ARXIV.2503.18929. URL `https://doi.org/10.48550/arXiv.2503.18929`.

[2] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL `http://arxiv.org/abs/1912.06680`.

[3] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish,

I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

[4] Z. Chen, A. May, R. Svirschevski, Y. Huang, M. Ryabinin, Z. Jia, and B. Chen. Sequoia: Scalable and robust speculative decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 129531–129563. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ea1f5f0878d43ff4fb8bf64ef4a2326c-Paper-Conference.pdf.

[5] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

[6] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

[7] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

[8] L. Espeholt, R. Marinier, P. Stanczyk, K. Wang, and M. Michalski. SEED RL: scalable and efficient deep-rl with accelerated central inference. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgvXlrKwH.

[9] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

[10] J. Hilton, K. Cobbe, and J. Schulman. Batch size-invariance for policy optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6ceb6c2150bbf46fd75528a6cd6be793-Abstract-Conference.html.

[11] J. Hu, X. Wu, W. Wang, Xianyu, D. Zhang, and Y. Cao. Openrlhf: An easy-to-use, scalable and high-performance RLHF framework. *CoRR*, abs/2405.11143, 2024. doi: 10.48550/ARXIV.2405.11143. URL https://doi.org/10.48550/arXiv.2405.11143.

[12] J. Hu, Y. Zhang, Q. Han, D. Jiang, X. Zhang, and H. Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *CoRR*, abs/2503.24290, 2025. doi: 10.48550/ARXIV.2503.24290. URL https://doi.org/10.48550/arXiv.2503.24290.

[13] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Dang, A. Yang, R. Men, F. Huang, X. Ren, X. Ren, J. Zhou, and J. Lin. Qwen2.5-coder technical report.

CoRR, abs/2409.12186, 2024. doi: 10.48550/ARXIV.2409.12186. URL https://doi.org/10.48550/arXiv.2409.12186.

[14] N. Jain, K. Han, A. Gu, W. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

[15] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

[16] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

[17] KimiTeam, A. Du, B. Gao, B. Xing, C. Jiang, C. Chen, C. Li, C. Xiao, C. Du, C. Liao, C. Tang, C. Wang, D. Zhang, E. Yuan, E. Lu, F. Tang, F. Sung, G. Wei, G. Lai, H. Guo, H. Zhu, H. Ding, H. Hu, H. Yang, H. Zhang, H. Yao, H. Zhao, H. Lu, H. Li, H. Yu, H. Gao, H. Zheng, H. Yuan, J. Chen, J. Guo, J. Su, J. Wang, J. Zhao, J. Zhang, J. Liu, J. Yan, J. Wu, L. Shi, L. Ye, L. Yu, M. Dong, N. Zhang, N. Ma, Q. Pan, Q. Gong, S. Liu, S. Ma, S. Wei, S. Cao, S. Huang, T. Jiang, W. Gao, W. Xiong, W. He, W. Huang, W. Wu, W. He, X. Wei, X. Jia, X. Wu, X. Xu, X. Zu, X. Zhou, X. Pan, Y. Charles, Y. Li, Y. Hu, Y. Liu, Y. Chen, Y. Wang, Y. Liu, Y. Qin, Y. Liu, Y. Yang, Y. Bao, Y. Du, Y. Wu, Y. Wang, Z. Zhou, Z. Wang, Z. Li, Z. Zhu, Z. Zhang, Z. Wang, Z. Yang, Z. Huang, Z. Huang, Z. Xu, and Z. Yang. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025. doi: 10.48550/ARXIV.2501.12599. URL https://doi.org/10.48550/arXiv.2501.12599.

[18] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In J. Flinn, M. I. Seltzer, P. Druschel, A. Kaufmann, and J. Mace, editors, *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

[19] K. Lei, Y. Jin, M. Zhai, K. Huang, H. Ye, and J. Zhai. PUZZLE: efficiently aligning large language models through light-weight context switch. In S. Bagchi and Y. Zhang, editors, *Proceedings of the 2024 USENIX Annual Technical Conference, USENIX ATC 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 127–140. USENIX Association, 2024. URL https://www.usenix.org/conference/atc24/presentation/lei.

[20] J. LI, E. Beeching, L. Tunstall, B. Lipkin, R. Soletskyi, S. C. Huang, K. Rasul, L. Yu, A. Jiang, Z. Shen, Z. Qin, B. Dong, L. Zhou, Y. Fleureau, G. Lample, and S. Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

[21] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. I. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3059–3068. PMLR, 2018. URL http://proceedings.mlr.press/v80/liang18b.html.

[22] B. Y. Lin, R. L. Bras, K. Richardson, A. Sabharwal, R. Poovendran, P. Clark, and Y. Choi. Zebralogic: On the scaling limits of llms for logical reasoning, 2025. URL https://arxiv.org/abs/2502.01100.

[23] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

[24] M. Luo, S. Tan, R. Huang, A. Patel, A. Ariyak, Q. Wu, X. Shi, R. Xin, C. Cai, M. Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025.

[25] M. Luo, S. Tan, J. Wong, X. Shi, W. Y. Tang, M. Roongta, C. Cai, J. Luo, L. E. Li, R. A. Popa, and I. Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013 2025. Notion Blog.

[26] Z. Mei, W. Fu, J. Gao, G. Wang, H. Zhang, and Y. Wu. SRL: scaling distributed reinforcement learning to over ten thousand cores. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=lajn1iROCu.

[27] Z. Mei, W. Fu, K. Li, G. Wang, H. Zhang, and Y. Wu. Realhf: Optimized RLHF training for large language models through parameter reallocation. *CoRR*, abs/2406.14088, 2024. doi: 10.48550/ARXIV.2406.14088. URL https://doi.org/10.48550/arXiv.2406.14088.

[28] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, Z. Zhang, R. Y. Y. Wong, A. Zhu, L. Yang, X. Shi, C. Shi, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703867. doi: 10.1145/3620666.3651335. URL https://doi.org/10.1145/3620666.3651335.

[29] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. J. Candès, and T. Hashimoto. s1: Simple test-time scaling. *CoRR*, abs/2501.19393, 2025. doi: 10.48550/ARXIV.2501.19393. URL https://doi.org/10.48550/arXiv.2501.19393.

[30] M. Noukhovitch, S. Huang, S. Xhonneux, A. Hosseini, R. Agarwal, and A. C. Courville. Asynchronous RLHF: faster and more efficient off-policy RL for language models. *CoRR*, abs/2410.18252, 2024. doi: 10.48550/ARXIV.2410.18252. URL https://doi.org/10.48550/arXiv.2410.18252.

[31] OpenAI, Sep 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

[32] OpenAI, Apr 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/.

[33] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[34] J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A. Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[36] L. Phan, A. Gatti, Z. Han, N. Li, J. Hu, H. Zhang, S. Shi, M. Choi, A. Agrawal, A. Chopra, A. Khoja, R. Kim, J. Hausenloy, O. Zhang, M. Mazeika, D. Anderson, T. Nguyen, M. Mahmood, F. Feng, S. Y. Feng, H. Zhao, M. Yu, V. Gangal, C. Zou, Z. Wang, J. P. Wang, P. Kumar, O. Pokutnyi, R. Gerbicz, S. Popov, J. Levin, M. Kazakov, J. Schmitt, G. Galgon, A. Sanchez,

Y. Lee, W. Yeadon, S. Sauers, M. Roth, C. Agu, S. Riis, F. Giska, S. Utpala, Z. Giboney, G. M. Goshu, J. of Arc Xavier, S. Crowson, M. M. Naiya, N. Burns, L. Finke, Z. Cheng, H. Park, F. Fournier-Facio, J. Wydallis, M. Nandor, A. Singh, T. Gehrunger, J. Cai, B. McCarty, D. Duclosel, J. Nam, J. Zampese, R. G. Hoerr, A. Bacho, G. A. Loume, A. Galal, H. Cao, A. C. Garretson, D. Sileo, Q. Ren, D. Cojoc, P. Arkhipov, U. Qazi, L. Li, S. Motwani, C. S. de Witt, E. Taylor, J. Veith, E. Singer, T. D. Hartman, P. Rissone, J. Jin, J. W. L. Shi, C. G. Willcocks, J. Robinson, A. Mikov, A. Prabhu, L. Tang, X. Alapont, J. L. Uro, K. Zhou, E. de Oliveira Santos, A. P. Maksimov, E. Vendrow, K. Zenitani, J. Guillod, Y. Li, J. Vendrow, V. Kuchkin, and N. Ze-An. Humanity's last exam. *CoRR*, abs/2501.14249, 2025. doi: 10.48550/ARXIV.2501.14249. URL https://doi.org/10.48550/arXiv.2501.14249.

[37] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887.

[38] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: memory optimizations toward training trillion parameter models. In C. Cuicchi, I. Qualters, and W. T. Kramer, editors, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM, 2020. doi: 10.1109/SC41405.2020.00024. URL https://doi.org/10.1109/SC41405.2020.00024.

[39] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

[40] N. L. Roux, M. G. Bellemare, J. Lebensold, A. Bergeron, J. Greaves, A. Fréchette, C. Pelletier, E. Thibodeau-Laufer, S. Tóth, and S. Work. Tapered off-policy REINFORCE: stable and efficient reinforcement learning for llms. *CoRR*, abs/2503.14286, 2025. doi: 10.48550/ARXIV.2503.14286. URL https://doi.org/10.48550/arXiv.2503.14286.

[41] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[43] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

[44] G. Shen, Z. Wang, O. Delalleau, J. Zeng, Y. Dong, D. Egert, S. Sun, J. J. Zhang, S. Jain, A. Taghibakhshi, M. S. Ausin, A. Aithal, and O. Kuchaiev. Nemo-aligner: Scalable toolkit for efficient model alignment. *CoRR*, abs/2405.01481, 2024. doi: 10.48550/ARXIV.2405.01481. URL https://doi.org/10.48550/arXiv.2405.01481.

[45] G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

[46] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019. URL http://arxiv.org/abs/1909.08053.

[47] C. V. Snell, J. Lee, K. Xu, and A. Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

[48] P. I. Team, S. Jaghouar, J. Mattern, J. M. Ong, J. Straube, M. Basra, A. Pazdera, K. Thaman, M. D. Ferrante, F. Gabriel, F. Obeid, K. Erdem, M. Keiblinger, and J. Hagemann. Intellect-2:

A reasoning model trained through globally decentralized reinforcement learning, 2025. URL https://arxiv.org/abs/2505.07291.

[49] P. I. Team, S. Jaghouar, J. Mattern, J. M. Ong, J. Straube, M. Basra, A. Pazdera, K. Thaman, M. D. Ferrante, F. Gabriel, F. Obeid, K. Erdem, M. Keiblinger, and J. Hagemann. Intellect-2: A reasoning model trained through globally decentralized reinforcement learning, 2025. URL https://arxiv.org/abs/2505.07291.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[51] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, Ç. Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019. doi: 10.1038/S41586-019-1724-Z. URL https://doi.org/10.1038/s41586-019-1724-z.

[52] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

[53] H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, and X. Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *CoRR*, abs/2405.14333, 2024. doi: 10.48550/ARXIV.2405.14333. URL https://doi.org/10.48550/arXiv.2405.14333.

[54] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Yang, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, X. Liu, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, Z. Guo, and Z. Fan. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024. doi: 10.48550/ARXIV.2407.10671. URL https://doi.org/10.48550/arXiv.2407.10671.

[55] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

[56] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin, K. Lu, M. Xue, R. Lin, T. Liu, X. Ren, and Z. Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

[57] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024. doi: 10.48550/ARXIV.2406.12045. URL https://doi.org/10.48550/arXiv.2406.12045.

[58] Z. Ye, L. Chen, R. Lai, W. Lin, Y. Zhang, S. Wang, T. Chen, B. Kasikci, V. Grover, A. Krishnamurthy, and L. Ceze. Flashinfer: Efficient and customizable attention engine for LLM inference serving. *CoRR*, abs/2501.01005, 2025. doi: 10.48550/ARXIV.2501.01005. URL https://doi.org/10.48550/arXiv.2501.01005.

[59] A. B. Yoo, M. A. Jette, and M. Grondona. SLURM: simple linux utility for resource management. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing, 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003, Revised Papers*, volume 2862 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003. doi: 10.1007/10968987_3. URL https://doi.org/10.1007/10968987_3.

[60] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W. Ma, Y. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

[61] Y. Yue, Y. Yuan, Q. Yu, X. Zuo, R. Zhu, W. Xu, J. Chen, C. Wang, T. Fan, Z. Du, X. Wei, X. Yu, G. Liu, J. Liu, L. Liu, H. Lin, Z. Lin, B. Ma, C. Zhang, M. Zhang, W. Zhang, H. Zhu, R. Zhang, X. Liu, M. Wang, Y. Wu, and L. Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025. URL https://arxiv.org/abs/2504.05118.

[62] Y. Zhao, A. Gu, R. Varma, L. Luo, C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

[63] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. W. Barrett, and Y. Sheng. Sglang: Efficient execution of structured language model programs. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

[64] Y. Zhong, Z. Zhang, X. Song, H. Hu, C. Jin, B. Wu, N. Chen, Y. Chen, Y. Zhou, C. Wan, H. Zhou, Y. Jiang, Y. Zhu, and D. Jiang. Streamrl: Scalable, heterogeneous, and elastic rl for llms with disaggregated stream generation, 2025. URL https://arxiv.org/abs/2504.15930.

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 main scope is in Section 2 and the main contribution is in Section 4. The abstract summarizes them.

   Guidelines:

   - The answer NA 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 NA 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 E

   Guidelines:

   - The answer NA 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: All assumptions have been illustrated in Proposition 1 and Section 2

Guidelines:

- The answer NA 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: We have discussed the model, dataset, and hyper-parameters we use in Appendix B and Section 6.

   Guidelines:

   - The answer NA 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: We have included code in the supplementary material. Datasets and models we used are all open-sourced.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix B

   Guidelines:

   - The answer NA 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: [No]

   Justification: We did not include error bars for large-scale end-to-end experiments because they are expensive to run. We present results within a single trial under the same fixed random seed across different settings.

   Guidelines:

   - The answer NA 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: In Appendix B and Section 6.

   Guidelines:

   - The answer NA 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: N/A

   Guidelines:

   - The answer NA 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: [No]

    Justification: This paper aims to optimize a training system, which has a limited social impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper uses datasets and models used by prior works.

    Guidelines:

    - The answer NA 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 original sources are all properly cited.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Reproducibility

The code of AReaL is available at `https://github.com/inclusionAI/AReaL/`. Datasets and base models in our experiments are all taken from the open-source community (see Appendix B). We used a fixed random seed of 1 across all experiments.

B Implementation Details

B.1 PPO Details

We disable the critic model and the reference model in PPO. The advantage estimation parameter λ in GAE and the RL discount factor γ are fixed at 1. The reward is 5 at the final token if the answer is correct and -5 otherwise. We additionally adopt advantage normalization across the global batch to stabilize the training. Other learning related hyperparameters and configurations can be found in Table 3.

Table 3: Training configurations and hyperparameters.

Training Configuration	
Batch size (number of prompts)	512
Random seed	1
PPO Parameters	
PPO Minibatches	4
Clipping ϵ	0.2
Advantage normalization	True
Discount factor γ	1.0
GAE λ	1.0
Optimizer Parameters	
Optimizer	Adam
Learning rate	2.0×10^{-5}
Weight decay	0.05
β_1	0.9
β_2	0.95
Adam ϵ	1×10^{-5}
Gradient norm clipping	1.0
Learning rate scheduler	constant
Warmup steps proportion	0.001
Precision Parameters	
Parameter dtype	fp16
KV cache dtype	fp16
Gradient dtype	fp32
Optimizer state dtype	fp32
Generation Parameters	
Answers per prompt	16
Temperature	1.0
Top-p	1.0
Top-k	-1
Max prompt length	1024
Min generation length	0
Max generation length	27648

B.2 Dataset Details

For the math task, we used the open-source data from DeepScaleR [25], For code training, we used the dataset released by DeepCoder [24]. All compared methods use the same dataset.

B.3 Dynamic Batching

Algorithm 1 Dynamic Batching

Require: Sequence lengths $S = \{s_1, s_2, \ldots, s_n\}$, maximum micro-batch capacity C, minimum number of micro-batches k_{min}
Ensure: Balanced partition of sequences into micro-batches with total length $\leq C$

```
Sort S in descending order
batches ← ∅
for all s ∈ S do
    if |batches| < k_min or no existing batch can fit s then
        Create new micro-batch containing sequence i
        batches.append({s})
    else
        Find all batches that can accommodate s
        Select the micro-batch with fewest sequences
    end if
end for
return batches
```

The dynamic batching algorithm is shown in Algorithm 1.

B.4 Baselines

In our experiments, we use the lastest version (main branch of verl repository, May 7, 2025) of verl [45] to evaluate the training throughput in Figure 4 and the training hours in Table 1. For most of the results, we use SGLang [63] v0.4.6 as generation backend and pytorch FSDP [62] as training backend. In a few cases where SGLang raises errors (experiments with 32B models or 64 nodes), we use vLLM [18] v0.8.4 as a substitution.

C Additional Results

C.1 Additional Evaluation Results

We evaluate the models trained with AReaL on more math and coding benchmarks, and list the results in Table 4 and Table 5, respectively.

Table 4: Results on math benchmarks.

Model	AIME24	AIME25	AMC23	MATH 500
1.5B basemodel	29.3	24.4	71.0	84.3
w/ Sync. AReaL	42.0	32.9	84.4	89.2
w/ AReaL	42.2	32.0	85.1	89.5
7B basemodel	54.3	41.7	89.5	92.8
w/ Sync. AReaL	63.0	50.0	93.2	94.2
w/ AReaL	63.1	47.3	93.6	94.3

C.2 Generalization Across Model Architectures

We conducted additional experiments using the DeepSeek-Distilled-Llama-8B model, which is a long-CoT model based on Llama 3.1 8B. We matched the experimental configuration with the Qwen 7B math model from Table 1, and the results are presented in Table 6.

Table 5: Results on coding benchmarks.

Model	LiveCodeBench v5	Codeforces	CodeContests
14B base model	53.4	1801/95.8%	32.0
Sync. AReaL 14B	56.7	1845/96.4%	37.0
AReaL 14B (ours)	58.1	1840/96.3%	35.9
32B base model	57.4	1839/96.3%	34.3
Sync. AReaL 32B	61.2	1911/96.9%	36.3
AReaL 32B (ours)	61.0	1889/96.7%	36.5

Table 6: Generalization results on DeepSeek-Distilled-Llama-8B across math benchmarks.

Model	AIME24	AMC23	MATH500	AIME25
DeepSeek-Distilled-Llama-8B	50.4	84.2	89.1	23.3
AReaL Fine-Tuned η =4	58.4	92.3	92.2	42.6
AReaL Fine-Tuned η =8	57.2	91.5	91.9	41.6

The results demonstrate that AReaL generalizes effectively across different model families.

C.3 Staleness-Throughput Trade-off with Small-Scale Academic Setups

We conducted a series of experiments using the DeepSeek-Distilled-Qwen-1.5B model with 8k context length and batch size 64×16 on 8 GPUs, testing various staleness values. The following table shows the experimental results. We observe that our preliminary conclusions from the large-scale setting (Table 2) generally align with findings using fewer GPUs.

Table 7: Staleness-throughput trade-off on small-scale academic setup.

Model	AIME24	AIME25	AMC23	MATH500	Throughput
DeepSeek-Distilled-Qwen-1.5B	29.3	24.4	71.0	84.3	-
AReaL Fine-Tuned η =0	31.7	26.1	78.9	86.7	27.1k
AReaL Fine-Tuned η =1	32.6	26.4	76.6	86.4	47.8k
AReaL Fine-Tuned η =2	32.4	26.7	76.0	86.6	47.8k
AReaL Fine-Tuned η =4	34.1	28.1	75.5	86.9	49.0k
AReaL Fine-Tuned η =8	29.9	23.2	76.1	86.1	51.5k
AReaL Fine-Tuned η =16	32.8	25.9	78.1	86.3	52.0k

C.4 Staleness-Throughput Trade-off with Different RL Algorithms

We conducted additional experiments using the RLOO advantage bootstrapping method. We trained 1.5B models in a local 8-GPU setting with 8k context length and various staleness values. The evaluation results in Table 8 demonstrate that RLOO exhibits slightly better tolerance to asynchronous training compared to vanilla PPO.

Table 8: Staleness-throughput trade-off using RLOO algorithm.

Model	AIME24	AIME25	AMC23	MATH500	Throughput
DeepSeek-Distilled-Qwen-1.5B	29.3	24.4	71.0	84.3	-
RLOO η =0	32.4	29.2	79.2	87.3	27.1k
RLOO η =1	32.9	26.0	76.4	87.7	47.8k
RLOO η =2	34.1	28.0	81.1	86.9	47.8k
RLOO η =4	32.9	27.9	76.0	87.0	49.0k
RLOO η =8	31.5	28.1	78.0	87.4	51.5k
RLOO η =16	32.7	27.7	78.4	87.4	52.0k

These results highlight an important direction for future research. AReaL modifies the PPO/GRPO algorithm because the importance sampling term naturally supports asynchronous off-policy training. Beyond PPO-based workflows, it would be valuable to investigate the asynchronous tolerance of REINFORCE-like and other off-policy algorithms.

D Proof of Proposition 1

Proposition 1. *For any sequence $(q, a_1, \ldots, a_H)$ generated by policies $(\pi_\theta, \ldots, \pi_{\theta+k})$ where $\pi_{\theta+i}$ produces tokens $(a_{t_i}, \ldots, a_{t_{i+1}})$, where $1 = t_0 < \cdots < t_{k+1} = H$, there exists a behavior policy π_{behav} such that the interrupted generation is equivalent to sampling entirely from π_{behav}.*

Proof. For question q, let $\mathcal{S}_t(q)$ denote states encountered at step t by the sequence of policies. Since $\mathcal{S}_{t_i}(q) \cap \mathcal{S}_{t_j}(q) = \emptyset$ for $i \neq j$, we can construct:

$$\pi_{\text{behav}}(\cdot|s) = \begin{cases} \pi_{\theta+j}(\cdot|s) & \text{if } t_j \leq t \leq t_{j+1} \text{ and } s \in \mathcal{S}_t(q) \\ \text{arbitrary} & \text{otherwise} \end{cases}$$

□

E Limitations and Future Work

Our work presents several limitations that suggest directions for future research. First, the ratio between inference and training devices could be further optimized for specific training setups. Additionally, this ratio might benefit from dynamic adjustment during training, particularly as context lengths typically increase when fine-tuning pre-trained base models. While we focused our evaluation on single-step mathematical and coding tasks, the AReaL architecture is not inherently limited to these domains. We leave the exploration of multi-turn interactions and agentic scenarios to future work.