# SortedRL: Accelerating RL Training for LLMs through Online Length-Aware Scheduling

**Yiqi Zhang** [†*1] **Huiqiang Jiang** [†2] **Xufang Luo** [†2] **Zhihe Yang** [*3] **Chengruidong Zhang** [2] **Yifei Shen** [2] **Dongsheng Li** [2] **Yuqing Yang** [2] **Lili Qiu** [2] **Yang You** [1]

## Abstract

Scaling reinforcement learning (RL) has shown strong promise for enhancing the reasoning abilities of LLMs, particularly in tasks requiring long chain-of-thought generation. However, RL training efficiency is often bottlenecked by the rollout phase, which can account for up to 70% of total training time when generating long trajectories (e.g., 16k tokens), due to slow autoregressive generation and synchronization overhead between rollout and policy updates. We propose *SortedRL*, an online length-aware scheduling strategy designed to address this bottleneck by improving rollout efficiency and maintaining training stability. SortedRL reorders rollout samples based on output lengths, prioritizing short samples forming groups for early updates. This enables large rollout batches, flexible update batches, and near on-policy micro-curriculum construction simultaneously. To further accelerate the pipeline, SortedRL incorporates a mechanism to control the degree of off-policy training through a cache-based mechanism, and is supported by a dedicated RL infrastructure that manages rollout and updates via a stateful controller and rollout buffer. Experiments using LLaMA-3.1-8B and Qwen-2.5-32B on diverse tasks, including AIME 24, Math 500, and GPQA, show that SortedRL reduces RL both training steps and bubble ratios by over 50%, while achieving equal or better performance.

## 1. Introduction

Large language models (LLMs) excel across a broad spectrum of tasks (Achiam et al., 2023; Yang et al., 2025; Gemini; Liu et al., 2024). Reinforcement learning (RL) has become a key tool for pushing those abilities further—especially on reasoning-heavy benchmarks such as complex maths (Hendrycks et al., 2021; He et al., 2024) and competition-level coding (Jain et al., 2025). State-of-the-art pipelines interleave two phases: (1) *rollout*, where the actor produces chain-of-thought (CoT) traces and final answers; (2) *update*, where rewards based on answer correctness drive policy improvement (Jaech et al., 2024; Guo et al., 2025; Seed et al., 2025).

We observe that better reasoning almost always comes with longer responses, yet long auto-regressive generation is painfully slow and hardware-inefficient. To reconcile sample-efficiency with compute efficiency, we introduce **SortedRL**. Its three pillars are:

1. **Online length-aware scheduling**: sort rollouts by predicted length and update on the short ones first, creating a micro-curriculum at no extra cost.
2. **Controlled off-policy radius**: cache unfinished samples and reuse them only within a tunable freshness window.
3. **Length-aware infrastructure**: a stateful controller and buffer that pipe data through the RL stages while squeezing bubbles out of the GPU timeline.

Across reasoning suites like OlympiadBench, AIME 2024, AMC 2023—we cut input usage by 41-53 % and still outperform baselines by up to 12.8 points, while slashing bubble ratio from $74\%$ to $< 6\%$.

## 2. Motivation and Preliminaries

### 2.1. Rollout as the Cost Sink

RL for LLMs generally has three stages: rollout, advantage estimation, and parameter updates. With modern 8B LLM and a 16 K-token cap, rollout alone can swallow $\sim 70\%$ of total training time (Yu et al., 2025). The culprit is the memory-bound, weight-and-KV-heavy nature of auto-regressive decoding—an issue that will only worsen for agentic or multi-turn setups (Wang et al., 2025).

[†] Equal contribution. [*]Work during internship at MSRA. [1]National University of Singapore [2]Microsoft Research Asia [3]CUHK. Correspondence to: Yiqi Zhang, Yang You <yiqi.zhang@u.nus.edu, dcsyouy@nus.edu.sg>.

*Work presented at the ES-FoMo Workshop at ICML 2025*, Vancouver, Canada. 

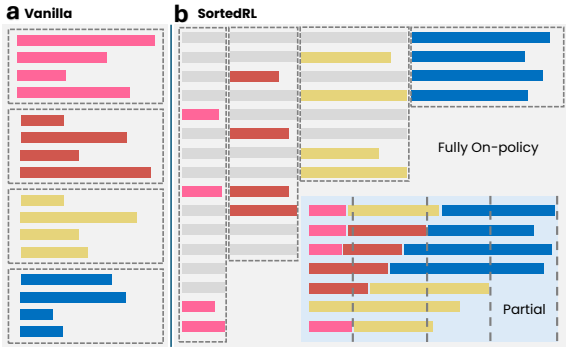



Figure 2: The SortedRL Scheme. **a** and **b** are imaginary timeline of baseline and SortedRL strategy. Samples in same batch are denoted in same color. Dotted lines and boxes indicates the harvest timing. For fully on-policy mode, the gray bars are discarded incomplete samples or non-scheduled prompts, while there is no discarded trajectories in partial mode.

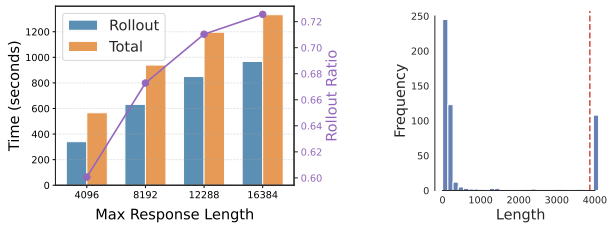



Figure 1: (a) Latency breakdown of RL training for LLMs. (b) Length distribution of sampled trajectories during rollout. Visualizations are based on DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) with a 4K maximum generation length.

### 2.2. Long-Tail Lengths → GPU Bubbles

Generation lengths follow a heavy-tailed law (Fig. 1b): in a batch of 512, 80 % finish within 3 K tokens but the slowest 20 % drag on to the limit. Because actor updates must wait for all trajectories in an on-policy batch, those stragglers force the rest of the GPU to idle—a phenomenon we measure as the *bubble ratio* (eq.1), where $Q$, $T$, $r_k$, $t_k$ denote running queue size, total elapsed time, running requests, and duration, respectively. Prior tricks like continuous batching (Yu et al., 2022) shine in inference but break here, and stage-fusion attempts (Zhong et al., 2025; Mei et al., 2025) lose steam as sequences grow longer.

$$\text{Bubble Ratio} = \frac{\sum (Q - r_k)\, \Delta t_k}{T\, Q}, \qquad (1)$$

SortedRL tackles *both* pain points by front-loading short samples and overlapping compute with generation progress, turning the long-tail from a liability into free curriculum.

## 3. SortedRL

Following the analysis in §2, we propose SortedRL to reduce the bubble ratio in the rollout stage and accelerate RL training for LLMs. SortedRL comprises three key components: 1) Online Length-Aware Scheduling, a length-aware batching mechanism is employed to minimize rollout bubbles by aligning samples with similar generation lengths within a batch. 2) Controllable Off-Policy Sampling, supports both on-policy and partial off-policy modes, enabling flexible trade-offs between stability and sample efficiency during training. 3) Co-Designed RL Infrastructure, includes a length-aware controller and a stateful rollout buffer, designed to coordinate length scheduling, buffer management, and model interaction efficiently.

### 3.1. Online Length-Aware Scheduling

To address the prolonged duration of the rollout stage and the high bubble ratio in RL training, we propose an online length-aware scheduling method. This approach dynamically batches samples with similar output lengths to minimize idle computation and reduce bubble ratios during rollout. To ensure token efficiency, we introduce the following key designs:

**Oversubscription and Early Termination.** To maximize hardware utilization, reduce bubble ratios during the rollout stage, and enable online awareness of generation length, our controller adopts an oversubscription strategy, feeding the rollout engine with more prompts than its maximum queue capacity in most iterations. This ensures that the rollout engine consistently operates at its optimal batch size, as captured by hardware runtime graphs (e.g., CUDA and HIP graphs), which is essential for maximizing the efficiency of JIT-compiled kernels.

In conjunction with oversubscription, the controller applies early termination based on batching-related thresholds. Once the condition is met, both completed and partially generated outputs are harvested. This mechanism effectively reduces idle time and minimizes computation bubbles during the rollout phase.

**Grouped Rollout.** While oversubscription and early termination improve hardware efficiency, they also introduce a side effect: longer generations are more likely to be interrupted during training, resulting in a bias toward shorter responses in the collected data. Although partial generations can be scavenged and resumed in the next rollout iteration, the resumed segments are inevitably off-policy, potentially affecting training stability.

To mitigate this, we organize prompts into groups of batches and enforce a cache-aware loading policy: no new prompts are loaded from the dataloader until all cached prompts have

been consumed. This strategy ensures that all prompts are fully processed within a bounded timespan, avoiding prompt starvation and maintaining balanced training dynamics.

**Selective Batching for Training.** Unlike the canonical RL pipeline, our controller can selectively batch ready trajectories and feed them to the trainer in a deterministic, ordered fashion. This is particularly important for algorithms such as Reinforce++, where batch-wise normalization can substantially impact training outcomes. For example, by sorting ready trajectories based on their rewards, the controller can form batches with similar reward distributions, enhancing stability during training.

SortedRL supports two switchable modes of operation: fully on-policy and partial mode. As illustrated in Fig. 2, the fully on-policy mode discards all partial responses after collecting a sufficient number of completed rollouts, only scavenging prompts from the terminated queue. In contrast, the partial mode caches incomplete responses and allows them to be resumed in the next iteration. This mode offers a middle ground between strict on-policy training and fully off-policy training with large rollout batches.

As a result, SortedRL naturally batches responses of similar lengths together. Since shorter responses are typically completed earlier, outputs within a time slice tend to be temporally clustered, leading to length-sorted batching as the rollout progresses. Additionally, shorter generations often receive higher rewards, as they are less likely to be penalized or clipped due to length constraints. Meanwhile, prompts completed later indicates that the prompt requires more intermediate reasoning to resolve. As a result, batches in SortedRL groups naturally constructs a micro-curriculum that has incremental difficulty. This sorting behavior can also substantially affects batch normalization, together contributing to improved token efficiency, which will be demonstrated and discussed in §4.

### 3.2. Co-designed RL Infrastructure

For SortedRL, we co-design a compute-efficient rollout controller and a stateful rollout buffer to maximize MFU, support agile algorithmic experimentation, and maintain intermediate stateful results across rollout iterations. The underlying infrastructure provides the following key capabilities:

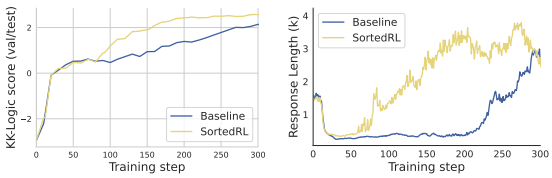
**Rollout State Manager.** The controller maintains a set of states, including: (1) unconsumed prompts from the dataloader, (2) scavenged response segments obtained upon rollout termination, and (3) the corresponding log probabilities for these segments. Scavenged segments can be concatenated with their original prompts to resume generation in subsequent iterations. Additionally, the stored log probability segments can be reused for importance sampling and serve as $\pi_{\text{old}}$ in Eq. 2 during policy updates.

**Stateful Rollout Buffer.** To support different modes in controllable off-policy training, we implement a stateful rollout buffer that stores intermediate results of partially generated trajectories. Specifically, each entry in the buffer includes: the prompt context, the current partial trajectory, a completion flag indicating whether the trajectory is finished, and a lifecycle indicator used to determine when the entry should be cleared. This design ensures that partially generated trajectories can be efficiently resumed or discarded based on training mode and resource constraints.

## 4. Results

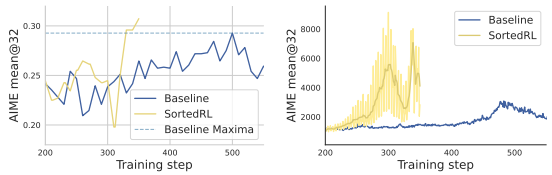
### 4.1. Experiment Setup

### 4.2. Logical Reasoning Tasks




(a) Evaluation metrics. (b) Mean response length.

Figure 3: LogicRL overall results.




(a) Evaluation metrics. (b) Mean response length.

Figure 4: Mathematical task overall results.

We first examine effects of our strategy on LogicRL dataset, on which we can obtain a stable learning pattern using vanilla methods. For better instruction following in logical games, we choose LLaMA-3.1-8B as starting point. Both baseline and SortedRL operates at a rollout batch size of 128 prompts (8 responses per prompt) and update batch size of 128 trajectories per step. For SortedRL, we chose fully on-policy mode to mitigate the off-policy update setting in hyperparameters.

The result are shown in Fig.3a. For both trials, the validation score underwent an sudden increase at initial 25 steps, where the model learnt to robustly capture the correct output format and avoided points deduction for format. At the same time, the mean response length also dropped to a minima. Subsequently, SortedRL rapidly improved in evaluation at around 80 steps, while baseline approach improves

Table 1: Evaluation results at the $4^{th}$ epoch. **Untuned** refers to checkpoint at $200^{th}$ step.

| | GSM8K | MATH 500 | Minerva | Olympiad | AIME (mean@32) | AMC (mean@32) |
|---|---|---|---|---|---|---|
| **SortedRL** | **94.84** | 62.00 | 25.00 | **37.39** | **29.58** | **75.16** |
| **Baseline** | 94.09 | **62.20** | 29.78 | 36.65 | 26.46 | 72.11 |
| **Untuned** | 93.03 | **62.20** | **30.88** | 34.87 | 24.79 | 69.69 |

at a slower pace. Until achieving comparable evaluation score (for instance, 2 points), baseline method lagged for around 3 epochs. Interestingly, the leading behavior is also exhibited in the response length pattern. In SortedRL, the model started to explore lengthy responses 150 steps earlier than baseline.

### 4.3. Math Reasoning Tasks

To explore the effects of SortedRL in more challenging tasks, we finetuned Qwen2.5-32B on DAPO-Math-17k dataset. This setting is substantially resource demanding than previous setting. Therefore, both our methods and baseline method starts at a 200-step checkpoint to prevent redundant computation, where the output format is consolidated. Meanwhile, we also enabled partial mode for optimal computational efficiency. Both methods operated at rollout and update batch size of 512.

As a result, shown in Fig.4a, SortedRL achieved 30.73% accuracy in AIME24 evaluation (average in 32 attempts per question) within 4 epochs. Baseline methods took another 4.5 epochs to achieve a comparable result. Similar pattern of early exploration on longer responses is also observed in this trial.

We took the checkpoint at the $4^{th}$ epoch to evaluate both model's performance on an ensemble of benchmarks. Results are shown in Tab.1. Compared to the training start point (i.e., 200 step checkpoint), both methods achieved improved score on majority of the benchmarks except MATH500 and Minerva Math. With same amount of input, SortedRL attained higher results on GSM8K, AIME24, and AMC23 benchmark. Especially for competition problems, our checkpoint largely outperforms baseline by 3.12% and 3.05% in accuracy.

### 4.4. Analysis

#### 4.4.1. Ablation Study

We conducted a set of ablation study (Fig.5) to pinpoint our key designs: In first experiment, we *disabled grouped rollout*, but preserved an oversubscription strategy, i.e., feeding a lot prompts and harvest the first few ready responses. In this way the rollout easily bias to shorter responses. Consequently, the performance capped at less than 1 in validation score and stopped improving.

Then, we investigated the effect of fully on-policy mode. In ablation settings, we sort the batch *post hoc*, after all responses are generated. Here, we set the rollout batch size to 512 prompts, which is the number of prompts that will be consumed within 1 group in fully on-policy SortedRL. As a result, the oldest trajectory in this method is 4 times farther away from policy in training than other strategies. In contrast, in our approach, the prompts are consumed in 4 separate iteration, and trainer gets freshly generated responses in each step. The validation score shows that even with batch sorting, the off-policiness is holding back token-efficiency.

#### 4.4.2. Throughputs

We examined the end-to-end rollout efficiency of our underlying infrastructure by testing throughput of the different methods under the workload of 512 samples in 4 separate batches with a maximum generation length of 8k. To avoid non-deterministic behavior in generation, we set the sampling parameters for each sample to let generation lengths be exactly the same as baseline.

The results (Fig.6) are 3987, 4289, and 5559 output tokens per second respectively for baseline, fully on-policy mode and partial mode. Equivalent to a speedup of 7.57% and 39.48% for fully on-policy mode and partial model, respectively. We define a bubble ratio in eq.1. Compared to baseline bubble ratio of 74%, on-policy and partial mode reduced the number to 5.81% and 3.37%.

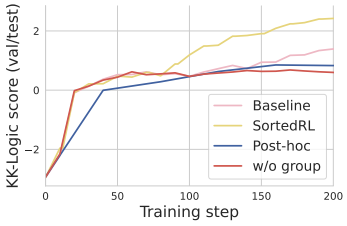



Figure 5: Ablation results.

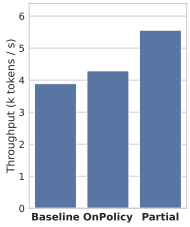



Figure 6: Rollout throughput.

## 5. Conclusion

*SortedRL* is an online, length-aware scheduling strategy that maximises hardware utilisation and boosts sample efficiency. Leveraging cache-based rollout control, it dynamically reorders training batches to form an on-the-fly micro-curriculum. In our tests, SortedRL cut the optimisation steps needed to match baseline performance by 40.7% on logical-reasoning tasks and 53.0% on mathematical benchmarks. With Qwen-2.5-32B, it pushed AIME24 accuracy past 30% a full 4.5 epochs earlier than the baseline. The sorted rollout also slashed computational bubbles from 74% to 5.8%, yielding nearly a 40% jump in rollout throughput. As a modular toolbox, SortedRL paves the way for richer research into RL rollout scheduling.

## References


Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B. S., Tumanov, A., and Ramjee, R. Taming throughput-latency tradeoff in llm inference with sarathi-serve, 2024. URL https://arxiv.org/abs/2403.02310.

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

Gemini. Context parallel api guide. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking. Accessed: May 9, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hu, J. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

Hu, J., Wu, X., Zhu, Z., Xianyu, Wang, W., Zhang, D., and Cao, Y. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Li, S., Liu, H., Bian, Z., Fang, J., Huang, H., Liu, Y., Wang, B., and You, Y. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, ICPP '23, pp. 766–775, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400708435. doi: 10.1145/3605573.3605613. URL https://doi.org/10.1145/3605573.3605613.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Mei, Z., Fu, W., Li, K., Wang, G., Zhang, H., and Wu, Y. Real: Efficient rlhf training of large language models with parameter reallocation. In *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. mlsys.org, 2025.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054, 2019. URL http://arxiv.org/abs/1910.02054.

Rajbhandari, S., Ruwase, O., Rasley, J., Smith, S., and He, Y. Zero-infinity: Breaking the GPU memory wall for extreme scale deep learning. *CoRR*, abs/2104.07857, 2021. URL https://arxiv.org/abs/2104.07857.

Ramamurthy, R., Ammanabrolu, P., Brantley, K., Hessel, J., Sifa, R., Bauckhage, C., Hajishirzi, H., and Choi, Y. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. 2022. URL https://arxiv.org/abs/2210.01241.

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

Seed, B., Yuan, Y., Yue, Y., Wang, M., Zuo, X., Chen, J., Yan, L., Xu, W., Zhang, C., Liu, X., et al. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

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.

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

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Shi, T., Wu, Y., Song, L., Zhou, T., and Zhao, J. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Wang, Z., Wang, K., Wang, Q., Zhang, P., Li, L., Yang, Z., Jin, X., Yu, K., Nguyen, M. N., Liu, L., Gottlieb, E., Lu, Y., Cho, K., Wu, J., Fei-Fei, L., Wang, L., Choi, Y., and Li, M. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Xie, C., Huang, Y., Zhang, C., Yu, D., Chen, X., Lin, B. Y., Li, B., Ghazi, B., and Kumar, R. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*, 2024.

Xie, T., Gao, Z., Ren, Q., Luo, H., Hong, Y., Dai, B., Zhou, J., Qiu, K., Wu, Z., and Luo, C. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2502.14768.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2407.10671*, 2025.

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, 2022.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

Zheng, L., Yin, L., Xie, Z., Sun, C. L., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., et al. Sglang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583, 2024.

Zhong, Y., Zhang, Z., Wu, B., Liu, S., Chen, Y., Wan, C., Hu, H., Xia, L., Ming, R., Zhu, Y., and Jin, X. Optimizing RLHF training for large language models with stage fusion. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pp. 489–503, Philadelphia, PA, April 2025. USENIX Association. ISBN 978-1-939133-46-5. URL https://www.usenix.org/conference/nsdi25/presentation/zhong.

# A. Backgrounds

## A.1. RL for LLMs

RL training for LLMs is a compute-intensive, multi-stage process characterized by heterogeneous components, unlike the uniformity in supervised finetuning. A typical RL pipeline consists of three key stages: (1) Rollout, where the actor model generates responses from input prompts; (2) Inference, where inference is performed using critic, reward, and reference models to compute values, rewards, and log-probabilities; and (3) Model update, where gradients are computed and applied. Scaling this pipeline has been shown to enhance LLM reasoning capabilities, especially in generating longer and more coherent Chain-of-Thoughts (CoTs).

## A.2. Proximal Policy Optimization (PPO) and Reinforce++

PPO (Schulman et al., 2017) and Reinforce++ (Hu, 2025) are both REINFORCE-based policy optimization methods. Specifically, they update the policy by maximizing the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D}, \\ o\sim\pi_{\theta_{\text{old}}}(\cdot|q)}} \left[ \min \left( \frac{\pi_\theta(o_t \mid q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q, o_{<t})} \widehat{A}_t,\ \text{clip}\left( \frac{\pi_\theta(o_t \mid q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q, o_{<t})}, 1-\varepsilon, 1+\varepsilon \right) \widehat{A}_t \right) \right], \tag{2}$$

where, $(q, a)$ denotes a question–answer pair sampled from the data distribution $\mathcal{D}$, $\varepsilon$ is the clipping range for the importance sampling ratio, and $\widehat{A}_t$ represents an estimator of the advantage at time step $t$. The computation and normalization of $\widehat{A}_t$ can vary across algorithms (Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025; Hu, 2025). For example, the advantage functions used in PPO and Reinforce++ are shown in Eq.(3) and Eq.(4), respectively.

$$\widehat{A}_{\text{PPO},t} = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \, \delta_{t+l}, \quad \delta_t = r_t + \gamma\, V_\psi(s_{t+1}) - V_\psi(s_t) \tag{3}$$

$$\widehat{A}_{\text{Reinforce++},t} = \frac{R_i - \mu_{\text{batch}}}{\sigma_{\text{batch}}} \tag{4}$$

# B. Limitations

Due to limited GPU resources, we conduct experiments on DAPO (Yu et al., 2025) with 4 epochs, and do not validate SortedRL under longer RL training schedules.

# C. Broader Impacts

SortedRL effectively accelerates RL training for LLMs, improving training efficiency in scenarios such as agentic reasoning, mathematical problem solving, and complex code understanding. By reducing rollout latency, it enables stronger reasoning capabilities under the same computational budget. Moreover, our length-aware scheduling is compatible with other rollout acceleration techniques, such as parallel decoding and speculative decoding, making it a versatile component for scalable RL training.

# D. Experiment Details

## D.1. Dataset

We evaluate our approach on two sets of data with distinct nature, and the ground truth data are suitable for rule-based evaluation:

First, a logical puzzle dataset, LogicRL (Xie et al., 2025). This dataset is composed of 5000 synthetic The Knights and Knaves game puzzles (Xie et al., 2024). An example can be checked in Fig.9, the game instructs players to deduce the roles of the characters mentioned in the statement. The training dataset is a mixture of 3 to 7 characters (i.e., different difficulties), with each difficulty accounting for 1000 samples. The samples are all shuffled during training. We spare 10% of data for evaluation.

Second is a mixed mathematical dataset, DAPO-Math-17k. This dataset contains a variety type of mathematical problems from the AoPS website. For easy and precise verification, the problems are transformed to expect an integer solution.

To evaluate the model's mathematical capability, we select 6 benchmarks following standard practice (Yu et al., 2025; Shi et al., 2025; Zeng et al., 2025): GSM8k (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), AIME 2024, and AMC 2023. For problems from competition, considering its relatively small amount, we collect 32 responses for each question and record accuracy (mean@32).

### D.2. Models

We select two scales of base model for two tasks. The model selection is determine by robust and reproducible baseline performance on specific downstream task. For example, we have observed that small model (Qwen-2.5-7B) is very limited in test-time scaling on math dataset, after an abrupt increment at the initial stage of RL training, which is possibly the process of learning the suitable format for rule-based verification, the model performance stopped improving (Fig.7b). Instead, Qwen-2.5-32B is free from such concern and exhibits robust and progressive improving pattern on training and evaluation metrics. As a result, for lightweight exploration on LogicRL, we employed LLaMA-3.1-8B-Instruct as base model, and Qwen-2.5-32B for mathematical problems.

### D.3. Compared Baseline

For logical Reasoning, following the original practice (Xie et al., 2025), we train LLaMA-3.1-8B-Instruct on LogicRL dataset with Reinforce++. We set the rollout prompt batch size to 128 and collect 8 responses from each prompt, then update with a trajectory batch size of 128.For mathematical tasks, we finetuned Qwen-2.5-32B on the DAPO-Math-17k dataset with PPO. The rollout prompt batch size, number of responses per prompt, update batch size are 512, 1, and 512, respectively. We adopt training tricks from DAPO (Yu et al., 2025) in both tasks including clip-higher, removing KL divergence. Meanwhile, we removed entropy loss for better stability.

### D.4. Implementation and Settings

Our rollout scheme is implemented as a integrated component of VeRL, an open-source RL training framework (Sheng et al., 2024). Experiments in this work are conducted with SGLang (Zheng et al., 2024) as rollout engine. Some patches are applied to facilitate better control of the rollout states and increase stability, including but not limited to: Incomplete request state retrieval, distributed communication timeout prevention, and platform-specific optimizations. Our training was conducted on a 2×8 AMD MI300X GPU cluster, equipped with 96-core Intel Xeon Platinum 8480C CPUs.

## E. Additional Analysis

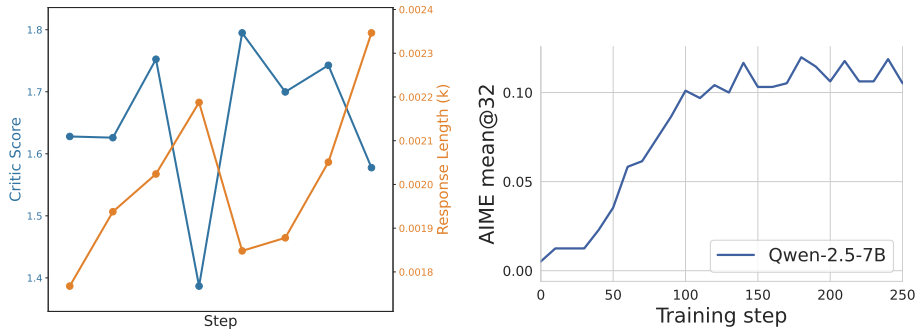
### E.1. Notes on math results

Interestingly, for Minerva Math benchmark, the accuracy drops as training progresses, regardless of training scheme. And MATH500 basically unchanged. This behavior is presumably originated from the training dataset characteristic: A notable characteristic of DAPO-Math-17k is that the solutions are all integers. This aligns with the robust improvements in integer-solution benchmarks like GSM8K, AIME24, and AMC23. On the flip side, MATH500, Minerva Math, and Olympiad all contains non-integer solutions that might be equations or fractions. Test-time scaling using integer-solution dataset might help build the reasoning process that facilitate answering the question, but contribute less on solving out-of-distribution tasks.

### E.2. Sensitivity to Grouping Size

SortedRL introduced a new hyperparameter, group size $n$. This refers to the number of prompt batches to be loaded by the SortedRL controller every time the prompt pool clears. Let rollout batch size be $b$, then the total number of prompts to be loaded into rollout buffer is denoted as $nb$. The controller does not load new prompts until every sample in current buffer are fed to the trainer.

Therefore, a large $n$ causes the rollout engine to keep generate answers that are clustered in same length. An extreme case is infinitely big $n$, at which the trainer only get short data. As shown in Fig.8, it fails to improve. Similar degradation is also identified on $n = 8$. In contrast, $n = 2$ results in a data distribution near the baseline approach, and consequently lead to a




(a) Response Length. (b) VAPO.

Figure 7: (a) A close-up look into mean scores and response lengths during training. (b)

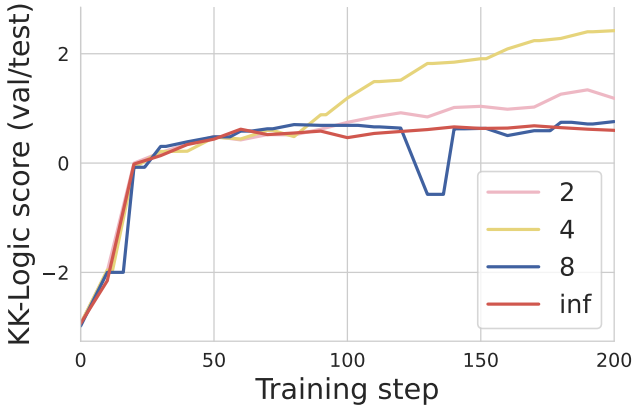



Figure 8: Result with different group size.

baseline-like curve.

### E.3. Complex Length Behaviour

Length is highly associated with the model performance. Reportedly in recent discoveries, reasoning capaility increment comes with increase response length. This aligns with the observations in our experiments (Fig.3b,4b). Given our length-sorted training schedule, we can have more insights into the role of response length in training. Fig.7a is a close-up look into two consecutive groups in SortedRL training. There is a clear short-short-short-long pattern in the rollout batches. This pattern forms a micro-curriculum for the training. The prompt remained in the buffer until clearing are generally questions require more reasoning steps to resolve.

Microscopically, longer responses tend to but not always have lower performance (Fig.7a). This can be partially explained by the fact that answers in longer sequences are more prone to be clipped. However, this low-score iteration is not harmful. Instead, the next iteration after long batch can achieve a higher score than previous short batches. Then the score keep drops as length increases until the next long batch concludes.

Macroscopically, we have some interesting observation in mathematical tasks. Just like local patterns mentioned earlier, the improvement of model performance are likely to show up on the falling edge of response length (310-330 steps, 340-350 steps of SortedRL; 480-550 in Baseline). Meanwhile, at comparable validation performance, SortedRL-tuned models has longer response length in both tasks.

## F. Related Work

### F.1. RL Training Systems

RLHF training frameworks have progressed from algorithm-centric libraries such as TRL (von Werra et al., 2020) and RL4LMs (Ramamurthy et al., 2022) to throughput-oriented systems like ColossalChat (Li et al., 2023), DeepSpeed-Chat

with ZeRO (Rajbhandari et al., 2019; 2021), and NeMo-Aligner (Shen et al., 2024), which scale to thousands of GPUs. The newest entrants—OpenRLHF (Hu et al., 2024) and VeRL/HybridFlow (Sheng et al., 2024)—simplify RLHF for non-experts and offer "parallel-native" execution across heterogeneous hardware, supporting 3-D, ZeRO, and FSDP parallelism out of the box. However, none of these frameworks yet provides online batch scheduling or fine-grained rollout control, two capabilities that our work introduces.

### F.2. LLM Generation Optimization

Modern RLHF rollouts piggy-back on high-throughput LLM-serving stacks: both OpenRLHF and VeRL use vLLM's PagedAttention for rapid KV-cache access (Kwon et al., 2023), while VeRL can switch to SGLang's Radix Attention, which pins shared cache segments to avoid recomputation (Zheng et al., 2024). These engines pair optimized CUDA/HIP kernels with graph capture, speculative decoding, continuous batching from Orca (Yu et al., 2022), and chunked prefill from Sarathi (Agrawal et al., 2024). Yet they still target low-latency, online inference with frozen weights, whereas RLHF demands high-throughput, batched "semi-offline" generation whose weights shift after every policy update—changes that invalidate cached kernels, disrupt KV-cache layouts, and create rollout-to-update "bubbles," while magnifying sensitivity to speed–accuracy trade-offs such as quantization. *SortedRL* closes this gap by shrinking those bubbles and markedly boosting rollout throughput.

## G. Dataset Example

```
Prompt
------
system
You are a helpful assistant. The assistant first thinks about the reasoning process
in the mind and then provides the user with the answer. The reasoning process and
answer are enclosed within <think> </think> and <answer> </answer> tags, respectively,
i.e., <think> reasoning process here </think><answer> answer here </answer>.
Now the user asks you to solve a logical reasoning problem. After thinking, when
you finally reach a conclusion, clearly state the identity of each character within
<answer> </answer> tags. i.e., <answer> (1) Zoey is a knight
(2)... </answer>.
<|im_end|>
<|im_start|>user
A very special island is inhabited only by knights and knaves. Knights always tell
the truth, and knaves always lie. You meet 3 inhabitants: Michael, Zoey, and Ethan.
Michael was heard saying, "Ethan is a knight if and only if Michael is a knight".
"Zoey is a knight or Ethan is a knight," Zoey mentioned. Ethan asserted:
"Michael is a knave if and only if Zoey is a knave".
So who is a knight and who is a knave?
<|im_end|>
<|im_start|>assistant
<think>assistant


Ground Truth
------
(1) Michael is a knight
(2) Zoey is a knight
(3) Ethan is a knight
```

Figure 9: Example (prompt, answer) pair for the LogicRL task.

```
Prompt
------
Solve the following math problem step by step. The last line of your
response should be of the form

    Answer: $Answer

(without quotes) where $Answer is the answer to the problem.

In triangle $ABC$, $\sin \angle A = \tfrac{4}{5}$ and $\angle A < 90^\circ$.
Let $D$ be a point outside triangle $ABC$ such that
$\angle BAD = \angle DAC$ and $\angle BDC = 90^\circ$.
Suppose that $AD = 1$ and that $\tfrac{BD}{CD} = \tfrac{3}{2}$.
If $AB + AC$ can be expressed in the form $\tfrac{a\sqrt{b}}{c}$ where
$a, b, c$ are pairwise relatively prime integers, find $a + b + c$.

Remember to put your answer on its own line after "Answer:".


Ground Truth
------
34
```

Figure 10: Example (prompt, answer) pair for the mathematical reasoning task.