SPEED-RL: Faster Training of Reasoning Models via Online Curriculum Learning

Anonymous Authors[1]

Abstract

Training large language models with reinforcement learning (RL) against verifiable rewards significantly enhances their reasoning abilities, yet remains computationally expensive due to inefficient uniform prompt sampling. We introduce SELECTIVE PROMPTING WITH EFFICIENT ESTIMATION OF DIFFICULTY (**SPEED**), an adaptive online RL curriculum that selectively chooses training examples of intermediate difficulty to maximize learning efficiency. Theoretically, we establish that intermediate-difficulty prompts improve the gradient estimator's signal-to-noise ratio, accelerating convergence. Empirically, our procedure leads to 2× to 6× faster training without degrading accuracy, requires no manual tuning, and integrates seamlessly into standard RL algorithms.

1. Introduction

Training large reasoning models (LLMs) commonly employs reinforcement learning (RL) techniques (Sutton et al., 1999) guided by automated verifiers (Ahmadian et al., 2024; Guo et al., 2025). However, this approach is computationally expensive, often surpassing the costs associated with conventional supervised fine-tuning. The substantial computational overhead constitutes a significant bottleneck for developing bespoke reasoning models for specific downstream tasks. Given the increasing importance of advanced reasoning skills across various domains, it is crucial to devise more compute-efficient strategies to train models with RL without sacrificing performance quality. This paper introduces and investigates data-driven techniques aimed at accelerating RL-based training of reasoning models.

Previous literature has demonstrated state-of-the-art reasoning performance by training LLMs on carefully curated datasets (Muennighoff et al., 2025; Ye et al., 2025; Li et al., 2025). However, these approaches predominantly depend on extensive human manual effort during data selection, limiting their scalability and practical flexibility. One attractive way to accelerate the training of reasoning models is to use *curriculum learning*, a training methodology that progressively organizes examples from simple to difficult. While curriculum learning has produced impressive results in some machine learning domains, it has yielded mixed or minimal improvements in others (Bengio et al., 2009; Soviany et al., 2022). On the other hand, the effectiveness of curriculum learning specifically integrated into RL training of LLMs for reasoning tasks remains poorly understood.

In this work, we propose a data-driven **online curriculum learning** strategy that dynamically prioritizes and selects informative training samples during RL training based on real-time estimates of difficulty. While similar insights have been leveraged by prior literature (Yang et al., 2024; Guo et al., 2025; Team et al., 2025; Foster and Foerster, 2025) as well as concurrent works (Bae et al., 2025; Lin et al., 2025; Yu et al., 2025; Xu et al., 2025; Cui et al., 2025), these methods have not yet fully realized the primary promise of online curriculum learning—namely, achieving a substantial reduction in *wall-clock* RL training time of reasoning models. To address this gap, this paper makes two primary contributions—one theoretical and one methodological—each reinforcing the other.

Theoretical contribution. We rigorously demonstrate that prompts of intermediate difficulty levels provide the strongest learning signal within RL frameworks. Specifically, we establish a novel theoretical link between the prompt pass rate and the *signal-to-noise ratio (SNR)* of stochastic gradient estimators—a critical factor for the convergence rate of optimization algorithms. Our analysis reveals that the gradient estimator's SNR significantly deteriorates for prompts with pass rates near 0% or 100%, where stochastic gradients become dominated by noise, making additional sampling meaningless. Crucially, our theoretical findings generalize to many widely-used policy gradient algorithms, such as REINFORCE(Sutton et al., 1999), REINFORCE++(Hu, 2025), RLOO(Ahmadian et al., 2024), GRPO(Shao et al., 2024), and PPO(Schulman et al., 2017) , and extend to domains beyond reasoning and language models. These insights directly inform our methodological approach.

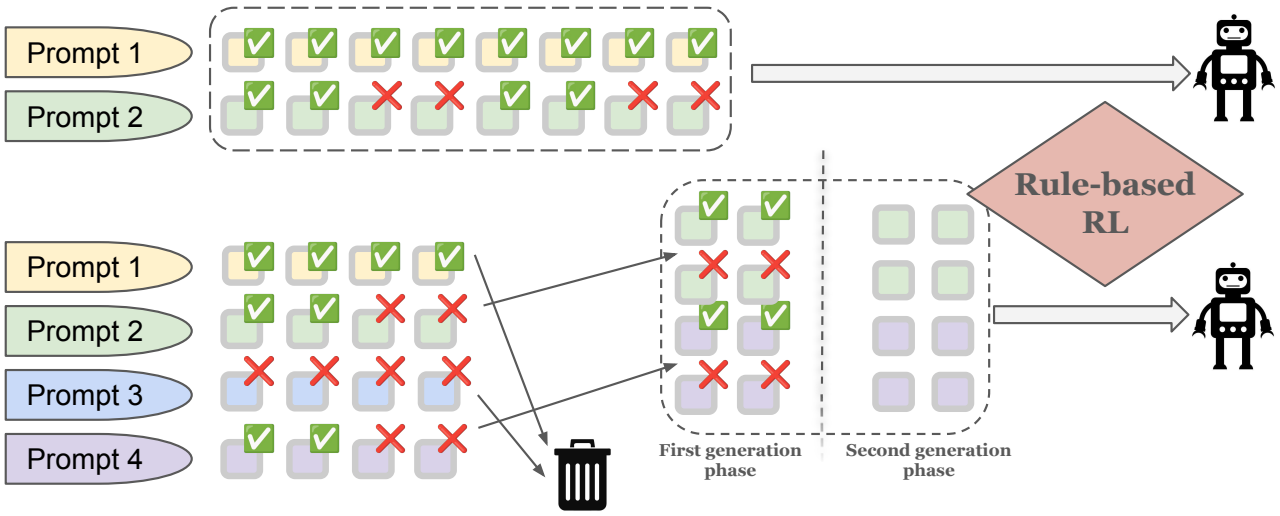

Figure 1: An overview of Selective Prompting with Efficient Estimation of Difficulty (**SPEED**).

Design principle and practical implementation. Guided by our theoretical insights, we introduce an efficient online curriculum learning framework that dynamically selects prompts at optimal difficulty levels for RL training. To achieve a lightweight online curriculum implementation, our procedure leverages the current state of the model itself to estimate the prompt difficulty. An instantiation of this idea is found in Foster and Foerster (2025), where many candidate responses per prompt are generated through full inference to estimate pass rates, and only informative prompts are selected for training. However, this naive method offers limited practical acceleration, as it does not address the heavy cost of inference, which is typically the most computationally demanding phase of RL training that dominates other costs.

On the contrary, our design reduces inference costs and enhances the quality of training signals for prompts selected during training, enabling more effective updates and faster convergence. Practically, we implement it by *screening the prompts' pass rates before performing expensive full inference for prompts with appropriate difficulty*. Specifically, we employ a lightweight statistical test to quickly estimate prompt difficulty, combined with an optimized pre-fetching mechanism that drastically reduces inference overhead. Experimental evaluations demonstrate substantial **wall-clock** speedups: our method **accelerates RL training by 2x to 6x** compared to baseline RL algorithms to achieve the same target performance across standard mathematical reasoning benchmarks. Furthermore, our curriculum method requires no manual data preprocessing or specialized auxiliary neural components, integrates seamlessly with prevalent RL algorithms (e.g., GRPO, REINFORCE, RLOO), and is broadly applicable to many tasks with binary-verifiable rewards.

2. Problem Setup

Let π_θ denote a language model parameterized by parameters θ. Given a prompt $x \in \mathcal{X}$, the model produces a response $y = (y^1, y^2, \dots) \in \mathcal{Y}$ auto-regressively. Formally, each token in the response sequence is generated according to the conditional probability:

$$y^{k+1} \sim \pi_\theta(\cdot \mid x, y^{\leq k}), \tag{1}$$

where we use $y^{\leq k}$ to refer to the first k tokens generated by the model. For reasoning-based tasks considered here, the model typically produces responses following a step-by-step "chain-of-thought" reasoning approach (Wei et al., 2022). A verification function $r(y)$ subsequently evaluates the correctness of the model's response based on the ground-truth answer, e.g.,

$$r(y) = \begin{cases} 1 & \text{if } y \text{ is correct,} \\ 0 & \text{if } y \text{ is incorrect.} \end{cases} \tag{2}$$

Our primary objective is to maximize the **pass rate**, defined as the average accuracy of the model π_θ across a distribution of prompts $\mathcal{D}_\mathcal{X}$. Explicitly, the optimization objective is defined as follows:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D}_\mathcal{X}} \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)], \tag{3}$$

where $\mathcal{D}_\mathcal{X}$ is a distribution over the prompt set. Taking the gradient of the objective (3) with respect to model parameters θ and employing a baseline (for variance reduction) (Sutton et al., 1999) leads to the well-known policy gradient formulation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{D}_\mathcal{X}} \mathbb{E}_{y\sim\pi_\theta(\cdot|x)} \left[\mathcal{A}(y) \nabla_\theta \log \pi_\theta(y \mid x)\right], \tag{4}$$

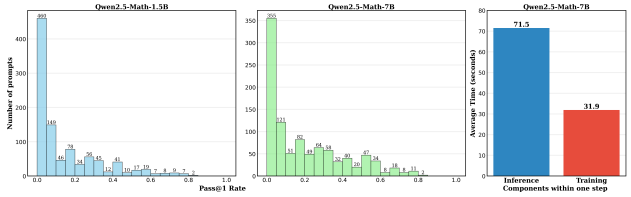

Figure 2: Left and middle: Pass rate distribution of 1000 samples in DAPO-17k evaluated by Qwen2.5-Math-1.5B (left) and Qwen2.5-Math-7B (middle). To evaluate the pass rates, we sample 50 responses per prompt. Right: Average per-step inference and training times while running RLOO on the Qwen2.5-Math-7B model.

where the advantage function $\mathcal{A}(y)$ is given by:

$$\mathcal{A}(y) = r(y) - \mathbb{E}_{y' \sim \pi_\theta(\cdot|x)}[r(y')]. \tag{5}$$

Here $\mathbb{E}_{y' \sim \pi_\theta(\cdot|x)}[r(y')]$ is the expected pass rate for prompt x. In practice, the policy gradient in equation (4) is estimated through Monte Carlo samples, yielding the classical *REINFORCE* algorithm (Sutton et al., 1999). Recent works have further modified this base policy gradient formulation to improve its stability, efficiency, and practicality. This has resulted in advanced methods such as REINFORCE++ (Hu, 2025), GRPO (Shao et al., 2024), PPO (Schulman et al., 2017), and RLOO (Ahmadian et al., 2024).

3. What Prompts Lead to Useful Training Signal?

Training datasets contains many "easy" and "hard" questions. Our work builds on the observation, shared by prior studies, that many training prompts are either too easy or too hard, and thus contribute a limited useful learning signal. Figure 2 makes this concrete: for 1000 prompts randomly drawn from the DAPO-17k dataset, we evaluate both Qwen 2.5-Math-1.5 B and Qwen 2.5-Math-7 B by sampling 50 completions for each prompt. Even with this generous sampling budget, there are still 340 and 258 prompts, respectively, with exactly zero pass rate. As we show in Section 5.1, these prompts—roughly one quarter to one third of the data—contribute no actionable learning signal.

In fact, in common reasoning benchmarks such as MATH (Hendrycks et al., 2021) or GSM8k (Cobbe et al., 2021), a large fraction of prompts are either trivially easy or effectively unsolvable, leading to pass-rate spikes at 0% or 100% (see Figure 3 of (Foster and Foerster, 2025)). (Schaeffer et al., 2025) also showed a heavy-tailed pass rate distribution for a subset of the MATH dataset (see their Figure 4). Additionally, as models evolve, many prompts become trivial for large models. DAPO (Yu et al., 2025) reports that late in training a 32B base model, over half of all prompts attain a 100% pass rate and provide no signals in the training (see their Figure 3b). These observations together motivate an *online, adaptive* filtering strategy: rather than fixing a static curriculum in advance, we continuously estimate each prompt's pass rate on-the-fly and retain only those whose difficulty remains in an intermediate range.

Effect of prompt difficulty on the policy gradient: An SNR perspective. It is natural to analyze how this skewed difficulty distribution influences the training signals, in particular, the resulting policy gradient (4). Intuitively, prompts that are either too simple or too difficult provide no informative feedback, as the model consistently succeeds or consistently fails, respectively. Formally, in both scenarios, the policy gradient for prompt x (Equation (4)) vanishes since the advantage function (Equation (5)) becomes zero:

$$\nabla_\theta J_x(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}\big[\underbrace{\mathcal{A}(y)}_{=0} \nabla_\theta \log \pi_\theta(y \mid x)\big] = 0 \tag{6}$$

if the pass rate is 0% or 100%. Thus, these prompts can be removed without affecting the gradient. This observation leads us to develop a more solid theory, which uses *information-theory* to measure the utility of a prompt. A natural measure quantifying the informativeness of a prompt is the **Signal-to-Noise Ratio (SNR)** of the policy gradient estimate. Formally speaking, we estimate (6) via empirical policy gradient:

$$\widehat{\nabla_\theta J_x}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \widehat{\mathcal{A}}(y_i) \cdot \nabla_\theta \log \pi_\theta(y_i \mid x), \tag{7}$$

where $y_1, ..., y_N \overset{i.i.d.}{\sim} \pi_\theta(\cdot \mid x)$. Here, $\widehat{\mathcal{A}}(y_i)$ is the advantage estimate for response y_i. For example, in our experiments, we adopt the RLOO estimator from Ahmadian et al. (2024):

$$\widehat{\mathcal{A}}(y_i) := r(y_i) - \frac{1}{N-1} \sum_{j \neq i} r(y_j). \tag{8}$$

With this definition, the empirical policy-gradient in (7) remains an unbiased estimator of the true gradient. The SNR is defined as the ratio between the squared norm of the gradient estimator and its variance:

$$\mathrm{SNR}(\theta) := \frac{\|\mathbb{E}\widehat{\nabla_\theta J_x}(\theta)\|^2}{\mathbb{E}\|\widehat{\nabla_\theta J_x}(\theta) - \mathbb{E}\widehat{\nabla_\theta J_x}(\theta)\|^2}. \tag{9}$$

Here, the SNR is a function of model parameter θ and prompt x. It quantifies the amount of information carried by a stochastic gradient, and governs the expected improvement of methods based on stochastic gradients—such as RLOO, PPO, REINFORCE, etc. For example, a simple connection between the SNR and the expected improvements is revealed in the following fact.

Algorithm 1 Selective Prompting with Efficient Estimation of Difficulty (SPEED)

Input: Prompt set $\mathcal{D}$; initial sample size N_{init}; continuation sample size N_{cont}; difficulty thresholds $P_{\text{low}}, P_{\text{high}}$; total training steps T; batch size B
for $t = 1$ **to** T **do**
  Initialize $\mathcal{D}_{\text{accepted}} \leftarrow \emptyset$
  repeat
    Sample prompt $x \in \mathcal{D}$
    Generate N_{init} responses from the current model
    $\textsc{PassRate}(x) \leftarrow \frac{1}{N_{\text{init}}} \sum_{i=1}^{N_{\text{init}}} \mathbb{I}(\text{response}_i \text{ is correct})$
    if $P_{\text{low}} < \textsc{PassRate}(x) < P_{\text{high}}$ **then**
      $\mathcal{D}_{\text{accepted}} \leftarrow \mathcal{D}_{\text{accepted}} \cup \{x\}$
    end if
  until $|\mathcal{D}_{\text{accepted}}| = B$
  for each $x \in \mathcal{D}_{\text{accepted}}$ **do**
    Generate an additional N_{cont} responses
  end for
  Perform one RL update step using all $N_{\text{init}} + N_{\text{cont}}$ responses for every $x \in \mathcal{D}_{\text{accepted}}$
end for

Fact 3.1. *Consider the one-step stochastic gradient update $\theta^+ = \theta + \widehat{\nabla_\theta J_x}(\theta)$, where the empirical policy gradient is unbiased. Assume $J_x(\theta)$ defined in (4) is 1-smooth. Then, one has*

$$\mathbb{E}[J_x(\theta^+)] - J_x(\theta) \geq \frac{1}{2}\|\nabla_\theta J_x(\theta)\|^2 \left(1 - \frac{1}{\text{SNR}(\theta)}\right), \tag{10}$$

where SNR is defined as (9).

This can be viewed as a natural consequence of the standard analysis of SGD on smooth functions (Duchi, 2018; Bernstein et al., 2018). We include the proof in Appendix A for completeness. Thus, if the SNR approaches zero, the variance completely overwhelms the signal, and negligible improvement is expected from a single update step. Conversely, as the SNR increases toward infinity, we recover the fast convergence behavior characteristic of deterministic gradient methods.

Connections between SNR and pass rate. Typically, common techniques to increase the SNR range from simply increasing the number of sampled rollouts to applying variance-reduction baselines (Sutton et al., 1999), which, in practical RL training, will be compute-heavy. In this work, instead, we select the most informative prompts to enhance the SNR during training, thus framing curriculum learning as a variance reduction tehcnique. To be precise, we establish an explicit link between the gradient estimator's SNR and the prompt pass rate, yielding the following fundamental result. We defer the proof to Appendix A.

Theorem 3.2 (Fundamental Connection between SNR and Pass Rate)**.** *Fix a prompt x. Let $\mathcal{P}(\theta)$ denote the pass rate of x under the current policy $(\pi_\theta(\cdot|x))$: $\mathcal{P}(\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[\mathbb{I}(r(y) = 1)]$. Then, the SNR of its stochastic gradient estimator (defined in (9)) satisfies*

$$\text{SNR} \leq f(\mathcal{P}(\theta)) := \left[\frac{(1-\mathcal{P}(\theta))^2}{\mathcal{P}(\theta)} + \frac{\mathcal{P}(\theta)^2}{1-\mathcal{P}(\theta)} - 1\right]^{-1} \tag{11}$$

Moreover, we have $\lim_{\mathcal{P}(\theta)\to 1} \text{SNR} = \lim_{\mathcal{P}(\theta)\to 0} \text{SNR} = 0, \arg\max_{\mathcal{P}(\theta)\in[0,1]} f(\mathcal{P}(\theta)) = \frac{1}{2}$.

This result is significant because it explicitly quantifies how much informative signal (relative to noise) a single training step provides as a direct function of pass rate. Crucially, Theorem 3.2 confirms an important intuition: prompts with very low ($\mathcal{P}(\theta) \approx 0\%$) or very high ($\mathcal{P}(\theta) \approx 100\%$) pass rates both yield vanishing SNR. Such prompts provide negligible useful training signal while potentially introducing detrimental variance in parameter updates. Therefore, the fundamental insight established by Theorem 3.2 is that optimal curricula must explicitly prioritize intermediate-difficulty prompts to maximize learning signal and effectiveness.

4. Algorithm

Algorithm design. In this section, we introduce SELECTIVE PROMPTING WITH EFFICIENT ESTIMATION OF DIFFICULTY (**SPEED**), an online curriculum learning method designed to feed training prompts at precisely the appropriate level of difficulty. Because the model proficiency evolves throughout training, the prompt difficulties must be continuously reassessed; this motivates our *adaptive, on-the-fly* curriculum design.

However, reliably identifying these prompts in a general and computationally efficient manner is challenging. Naively

estimating prompt difficulties by generating multiple responses per prompt (as studied recently in Foster and Foerster (2025)) and computing the pass rates quickly becomes computationally prohibitive, as inference usually dominates RL training time. As demonstrated in Figure 2, inference time for methods like RLOO often exceeds the actual time spent on gradient updates by a factor of two, even for moderately lengthy responses. This imbalance arises primarily due to the autoregressive nature of LLM inference and significantly slows down the RL training.

To address this, we propose a two-phase approach—illustrated in Algorithm 1—that efficiently filters and prioritizes prompts. Suppose the underlying RL method generates N rollouts per prompt (typically, $N = 16$ to 64 (Liu et al., 2024; Guo et al., 2025; Yu et al., 2025)). We split this into an *initial inference phase* ($N_{\text{init}} \approx 4$–$8 \ll N$) and *a continuation phase* (N_{cont}), where $N_{\text{init}} + N_{\text{cont}} = N$. First, a limited number of responses per prompt are generated to form a coarse pass-rate estimate. We then select prompts whose estimated pass rates lie distinctly away from trivial extremes (0% or 100%), defining these as **qualified prompts**. In the second phase, we produce the remaining responses (N_{cont}) only for these prompts, thus significantly reducing unnecessary inference. This selective two-phase inference scheme is compatible with common RL algorithms, such as GRPO, PPO, RLOO, and REINFORCE and it recovers them for appropriate choice of the hyperparameters [1].

Efficient implementation. Our algorithm involves two inference phases, which in a naive implementation would separately call the inference engine (e.g., vLLM (Kwon et al., 2023)) twice, increasing computational overhead. Since vLLM features efficient internal batching mechanisms(e.g., continuous batching and chunked prefill), performing two separate inference calls is slower than a single combined call.

To mitigate this overhead, we design an efficient prefetching mechanism. While performing the continued generation phase for accepted prompts, we simultaneously prefetch a new batch of prompts and rollout initial generations for these prompts. Responses for the new prompts are immediately evaluated for pass rates, while responses for accepted prompts are stored in the data buffer for training. This combined inference significantly reduces overhead and takes one inference call per generation step on average. Although responses for a prompt might be generated by slightly different model versions, as there may be an intermediate RL training step between the two generation steps, we empirically find no performance degradation, confirming the practical effectiveness of this approach.

Moreover, standard RL training methods typically operate with fixed batch sizes, meaning the number of prompts processed in each RL update step is predetermined. However, this fixed-batch restriction conflicts with the rejection-sampling nature of our proposed curriculum algorithm, for which the number of prompts accepted for training dynamically varies and may not exactly match the required batch size. Previous solutions either handle inference and training asynchronously (Xu et al., 2025) or resort to repetitive inference passes before each training step, discarding surplus prompts afterward (Yu et al., 2025). In contrast, we introduce a simple data buffer to temporarily store prompts not immediately utilized due to batch-size constraints. This strategy enables larger inference batches without data wastage, significantly increasing computational efficiency. While buffering introduces somewhat more off-policy training, our experimental results clearly demonstrate that this data-buffer approach markedly improves computational efficiency without compromising performance.

The full detailed algorithm combining the data buffer and pre-fetching mechanism is described in **??**. Moreover, the efficient implementation is compatible with more RL frameworks, especially those that require asynchronous or multi-stage training.

5. Experiments

In this section, we evaluate the performance of **SPEED**. We first describe our experimental setup, including the models, datasets, baseline methods, and evaluation metrics. We then present our primary results and discussions in Section 5.1.

Training setup. Our experiments use Qwen2.5-Math-1.5B and Qwen2.5-Math-7B models (Yang et al., 2024). We integrate SPEED with two rule-based RL methods: RLOO (Ahmadian et al., 2024) and DAPO (Yu et al., 2025). DAPO serves as an important baseline for curriculum learning as it filters all prompts with 0% or 100% pass rates after generating all responses. While our evaluation focuses on these two algorithms, our approach is broadly applicable to any rule-based RL method.

We train the models using three datasets: NuminaMath (Li et al., 2024), DAPO-17k (Yu et al., 2025), and DeepScaleR (Luo et al., 2025). NuminaMath originally contains 860k prompts, ranging from simpler GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) questions to challenging competition-level problems. We filter out proof-based questions and keep only problems with integer-valued solutions, which leaves us with 220k prompts. The DAPO-17k dataset consists of 17k integer-answer prompts, of which we reserve 1k as a held-out test set and use the remain-

[1] More precisely, when $P_{\text{low}} = -\infty, P_{\text{high}} = +\infty, N_{\text{init}} = N, N_{\text{cont}} = 0$.

Table 1: The wall-clock training hours needed for each model to reach a target accuracy on its respective benchmark. For the Qwen-Math-1.5B model, the targets are 0.30 on DAPO-1k, 0.70 on MATH500, 0.4 on AMC2023, and 0.10 on AIME. For the larger Qwen2.5-Math-7B model, the thresholds increase to 0.45, 0.80, 0.55, and 0.18 for the same datasets, respectively. Validation time and checkpoint-saving overhead are excluded. Values in parentheses give the corresponding wall-clock speed-ups. † means the performance has not reached the threshold during the entire training process.

Model Size	Training Data	Algorithm	DAPO-1k	MATH500	AMC2023	AIME
Math-1.5B	**NuminaMath**	RLOO	25.9	13.6	4.7	13.6
		SPEED-RLOO	7.6 (**3.4 x**)	3.3 (**4.1 x**)	2.8 (**1.7 x**)	6.4 (**2.1 x**)
		DAPO	†	18.0	10.0	16.7
		SPEED-DAPO	11.6 (†)	3.9 (**4.6 x**)	3.4 (**2.9 x**)	10.4 (**1.6 x**)
	DAPO-17k	RLOO	†	13.8	8.0	12.8
		SPEED-RLOO	4.4 (†)	4.8 (**2.9 x**)	2.2 (**3.6 x**)	4.8 (**2.7 x**)
Math-7B	**DAPO-17k**	RLOO	13.5	12.7	4.3	7.8
		SPEED-RLOO	3.6 (**3.8 x**)	4.3 (**3.0 x**)	1.8 (**2.4 x**)	3.0 (**2.6 x**)
		DAPO	12.1	21.8	7.6	7.6
		SPEED-DAPO	5.0 (**2.4 x**)	6.2 (**3.5 x**)	2.2 (**3.5 x**)	2.8 (**2.7 x**)
	DeepScaleR	RLOO	12.6	11.1	5.6	9.7
		SPEED-RLOO	2.9 (**4.3 x**)	2.9 (**3.8 x**)	1.1 (**5.1 x**)	1.6 (**6.1 x**)
		DAPO	17.1	16.2	7.5	11.1
		SPEED-DAPO	15.1 (**1.1 x**)	5.7 (**2.8 x**)	2.2 (**3.4 x**)	2.2 (**5.0 x**)

ing prompts for training. DeepScaleR contains approximately 400k training examples derived from past AIME (up to 2023) and AMC (prior to 2023) competitions. Besides the held-out test set in DAPO-17k, we evaluate the models' performance on four additional standard mathematical reasoning datasets: MATH500 (Lightman et al., 2023), AIME2024 (AIM, 2024), AIME2025 (AIM, 2024), and AMC2023 (AMC, 2023).

5.1. Results

Efficiency evaluation. We measure SPEED's efficiency improvements by comparing the relative wall-clock time needed to reach specific accuracy targets. When calculating the training time, we include every stage in the RL training except the time for validation and saving checkpoints. To ensure consistency, all experiments use a single node equipped with four NVIDIA GH200 GPUs (with 96GB of GPU memory and 120GB CPU memory each). Our implementation relies on the VeRL framework (Sheng et al., 2024). Unless otherwise specified, the training batch size (number of prompts) is set to 16, and the generation batch size is 64 for SPEED variants. For vanilla DAPO and SPEED-DAPO, we set $\varepsilon_{\text{low}} = 0.2, \varepsilon_{\text{high}} = 0.28$. In every experiment, we apply a learning rate of 10^{-6} with a warmup period of 10 steps and a weight decay of 0.1. Baselines generate $N = 24$ responses per prompt, and SPEED-RL variants use a combined $N_{\text{init}} + N_{\text{cont}} = 24$ generations.

Figure 3 and Table 1 illustrate that SPEED significantly enhances training efficiency. Integrated with rule-based RL algorithms RLOO and DAPO, SPEED achieves target validation accuracies 2–6 times faster compared with baseline RL algorithms across nearly all benchmarks and experimental runs. For instance, on DAPO-1k, Qwen2.5-Math-7B reaches a validation accuracy of 0.45 in 7.6 hours with SPEED-RLOO, whereas vanilla RLOO requires approximately 3.4 times longer. Although specific speedup values vary by dataset and target accuracy, our results consistently demonstrate substantial efficiency improvements across multiple setups.

Informativeness measures. To understand why SPEED improves efficiency, we examine the informativeness of gradients produced during training. As depicted in Figure 4, SPEED-RLOO consistently maintains training accuracies much closer to 0.5 compared to vanilla RLOO, particularly in early training stages. According to our theoretical analysis (Theorem 3.2), prompts with pass rates close to 0.5 generally yield higher SNR, which enhances training efficiency. Additionally, gradient norms from SPEED-RLOO are substantially larger than those from baseline methods, aligning well with our theoretical predictions discussed in Section 3.

Effect of N_{init}. The initial inference stage generation count (N_{init}) is the only additional hyperparameter introduced by our method, and setting $N_{\text{init}} = N$ and $N_{\text{cont}} = 0$ recovers the original RL method. A larger N_{init} increases the likelihood of selecting prompts with more extreme pass rates,

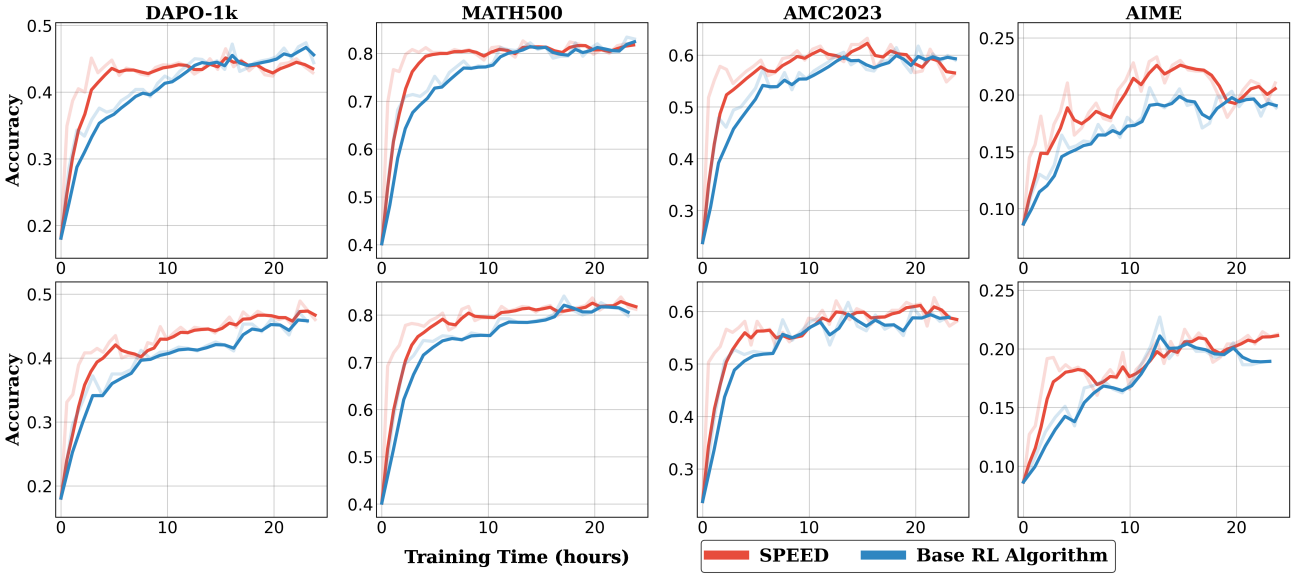

Figure 3: Validation accuracy on various mathematical reasoning benchmarks for SPEED-variants of RL algorithms, and base RL algorithms. Top: RLOO versus SPEED-RLOO; bottom: DAPO versus SPEED-DAPO. The initial model used is Qwen2.5-Math-7B, trained on the DeepScaleR dataset. The lighter curves represent raw accuracy results, while the bold curves indicate smoothed results obtained via an exponential moving average.

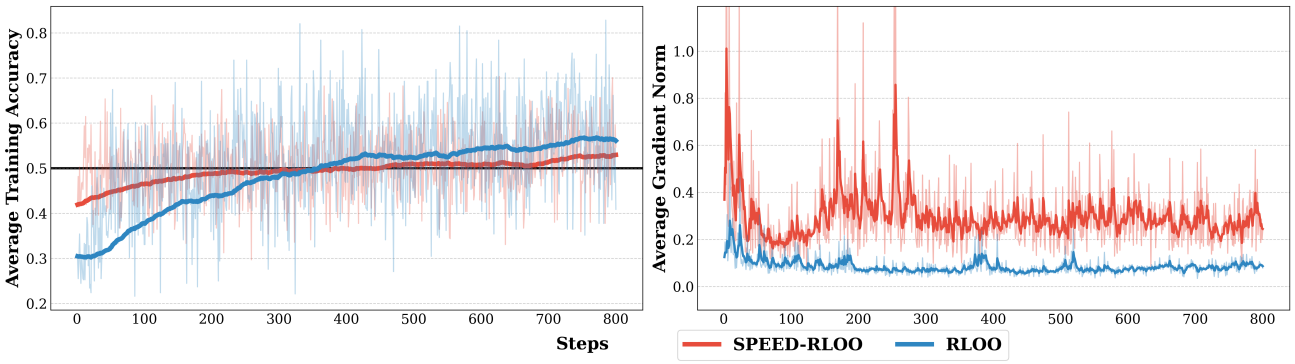

Figure 4: Average training accuracy (left) and gradient norm (right) comparison between RLOO and SPEED-RLOO during training of Qwen2.5-Math-7B. For the SPEED variants, the reported accuracies on the training set are calculated exclusively using the qualified prompts that are selected in the actual training process.

potentially reducing gradient informativeness. Figure 5 compares setups with $N_{\text{init}} = 4, 6, 8$ under identical conditions. Results show that larger N_{init} values lead to smaller average gradient norms and push training accuracies away from 0.5. This aligns with our theoretical insights that the prompts with pass rates near 0.5 can provide stronger learning signals. As a result, increasing N_{init} tends to slow down the performance rise and reduce the efficiency improvements compared to baselines.

6. Additional Related Works

Large reasoning models and reinforcement learning. Large language models (LLMs) have achieved remarkable performance on mathematical reasoning and code generation (Achiam et al., 2023; Yang et al., 2024; Jaech et al., 2024; Hurst et al., 2024; Lambert et al., 2024; Guo et al., 2025; Team et al., 2025). Fine-tuning these pretrained models often relies on reinforcement learning (RL) methods such as Proximal Policy Optimization (PPO) (Schulman et al., 2017; Hu et al., 2025). However, PPO updates can be computationally expensive and prone to reward hacking, limiting rapid iteration and deployment. To address these issues, several rule-based RL variants have been proposed. DeepSeek, for example, introduces Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Guo et al., 2025). Other extensions—such as REMAX (Li et al., 2023), RLOO (Ahmadian et al., 2024), and REINFORCE++ (Hu, 2025; Xie et al., 2025)—further demonstrate the benefits of rule-based RL.

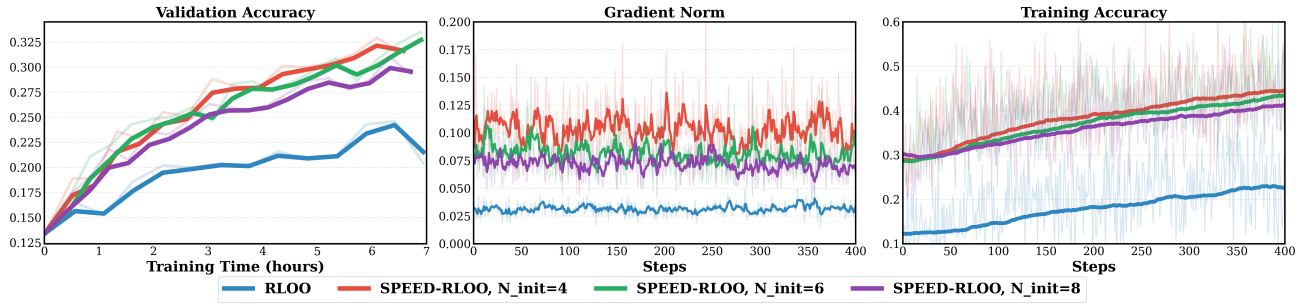

Figure 5: The validation accuracy on DAPO-1k (left), the average gradient norm (middle), and the average training accuracy (right) of RLOO and SPEED-RLOO with different N_{init}. We train Qwen2.5-Math-1.5B on the training split of DAPO-17k.

Difficulty-based curriculum learning. Training large reasoning models is often computationally expensive. *Curriculum learning* arranges training examples by increasing difficulty to guide model learning from moderately difficult examples (Bengio et al., 2009). Several open-source LLMs (e.g., Qwen2.5 (Yang et al., 2024), Kimi-1.5 (Team et al., 2025)) mention curricula without publishing details. Some *static* methods sort data offline—using pass-rate estimates (Wen et al., 2025), human difficulty labels (Lee et al., 2023), or software metrics for coding tasks (Naïr et al., 2024)—and then apply sequential supervised or RL fine-tuning. Recently, (Shi et al., 2025) estimates the question difficulty offline via the pass rates or from more capable models, and adaptively updates the target difficulty to select proper data in the training stage. In contrast, our method estimates prompt difficulty on the fly, yielding more accurate and timely example selection. Recent work on *online filtering* adapts the curricula to the model's current performance. DAPO (Yu et al., 2025) dynamically samples prompts and discards those with uniformly correct or incorrect responses. Bae et al. (2025); Lin et al. (2025); Meng et al. (2025) further restrict training to prompts with moderate pass rates. (Foster and Foerster, 2025) select the prompts with maximal reward variance. (Xu et al., 2025) adopts a similar idea and further decouples the inference and training phases to boost the efficiency. (Cui et al., 2025) uses a separately trained process reward model to gauge difficulty. They estimate the question difficulty via the pass rate after all responses are generated. Unlike these methods, we use a lightweight hypothetical generation step to infer difficulty, reducing compute and boosting the inference efficiency.

More methods for efficient reasoning. Beyond curriculum learning, researchers have explored data selection and inference optimizations for reasoning models. Curating high-quality chain-of-thought data can boost training efficiency (Muennighoff et al., 2025; Ye et al., 2025; Li et al., 2025), and token-level filtering can further reduce cost (Lin et al., 2024b). Another class of methods is based on early stopping or rejection sampling, such as RAFT (Dong et al., 2023) and speculative rejection (Sun et al., 2024). Our method effectively combines early stopping with difficulty filtering. Other approaches compresses chain-of-thoughts via prompt engineering (Han et al., 2024; Nayab et al., 2024), conditional training (Deng et al., 2024; Kang et al., 2025) or RL (Arora and Zanette, 2025; Fatemi et al., 2025). Additionally, efficient serving systems like vLLM (Kwon et al., 2023), speculative decoding (Leviathan et al., 2023; Liu et al., 2023), weight pruning (Liu et al., 2018), and quantization (Lin et al., 2024a) further cut runtime and memory requirements. These methods are orthogonal to our proposed algorithm and can be seamlessly combined with our method.

7. Conclusion and Future Directions

In this paper, we introduce **SPEED**, a method to accelerate the rule-based RL training of large reasoning models via online curriculum learning. By adaptively prioritizing prompts of intermediate difficulty, estimated by the pass rates over initially generated responses, SPEED selects prompts at the right level of difficulty on the fly to enhance the gradient informativeness. Our theory also shows that moderately difficult prompts can maximize the upper bound of Signal-to-Noise Ratio. Experiments demonstrated that SPEED significantly accelerates training, achieving between two to six times speedups across various datasets and tasks.

Future Directions. We identify several promising directions for future research:

- Although SPEED accelerates training efficiency, it does not necessarily enhance peak performance. Future work could explore methods that achieve rapid initial learning and superior final performance simultaneously.
- Our curriculum selects prompts based on difficulty estimates from the pass rates. Future research could explore alternative criteria for adaptive data selection. For example, for value-based RL algorithms, the value function may provide an efficient difficulty estimation.
- Future research could integrate reward evaluation directly into existing efficient inference servings like vLLM, allowing immediate, on-the-fly prompt filtering, which would further enhance the efficiency of training large reasoning models.

References

American mathematics competitions (amc 10/12), 2023, February 2023. URL https://artofproblemsolving.com/wiki/index.php/AMC_12_Problems_and_Solutions. Problems and solutions.

American invitational mathematics examination (aime) i and ii, 2024, February 2024. Problems and solutions.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, 2025.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

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

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

John C Duchi. Introductory lectures on stochastic optimization. *The mathematics of data*, 25:99–186, 2018.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Thomas Foster and Jakob Foerster. Learning to reason at the frontier of learnability. *arXiv preprint arXiv:2502.12272*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Bruce W Lee, Hyunsoo Cho, and Kang Min Yoo. Instruction tuning with human curriculum. *arXiv preprint arXiv:2310.09518*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024a.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*, 2024b.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Marwa Naïr, Kamel Yamani, Lynda Said Lhadj, and Riyadh Baghdadi. Curriculum learning for small code language models. *arXiv preprint arXiv:2407.10194*, 2024.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*, 2024.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? *arXiv preprint arXiv:2502.17578*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

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

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*, 2024.

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

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

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

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

A. Omitted proof in Section 3

Proof of Fact 3.1. From the definition of $1-$smoothness, we know $-J_x(\theta)$ is 1-smooth if $J_x(\theta)$ is 1-smooth. We have

$$-J_x(\theta+v) \le -J_x(\theta) + \nabla(-J_x(\theta))^\top v + \frac{1}{2}\|v\|^2$$

for any vector $v \in \mathbb{R}^d$ and parameter $\theta \in \mathbb{R}^d$. This implies

$$J_x(\theta+v) \;\ge\; J_x(\theta) + \nabla J_x(\theta)^\top v - \frac{1}{2}\|v\|^2.$$

Let $\widehat{g} := \widehat{\nabla_\theta J_x}(\theta)$ be an unbiased stochastic gradient so that $\mathbb{E}[\widehat{g}] = \nabla J_x(\theta)$. Applying this with $v = \widehat{g}$ and taking expectation,

$$\begin{aligned}
\mathbb{E}\big[J_x(\theta^+)\big] - J_x(\theta) &\ge \nabla J_x(\theta)^\top \mathbb{E}[\widehat{g}] - \frac{1}{2}\,\mathbb{E}\big[\|\widehat{g}\|^2\big] = \|\nabla J_x(\theta)\|^2 - \frac{1}{2}\Big(\mathbb{E}\|\widehat{g} - \mathbb{E}\widehat{g}\|^2 + \|\mathbb{E}\widehat{g}\|^2\Big) \\
&= \frac{1}{2}\,\|\nabla J_x(\theta)\|^2 - \frac{1}{2}\mathbb{E}\|\widehat{g} - \mathbb{E}\widehat{g}\|^2 \\
&= \frac{1}{2}\,\|\nabla J_x(\theta)\|^2 \left(1 - \frac{\mathbb{E}\|\widehat{g} - \mathbb{E}\widehat{g}\|^2}{\|\nabla J_x(\theta)\|^2}\right).
\end{aligned}$$

Invoking the definition of the Signal-to-Noise ratio in (9), we complete the proof. □

Now let's prove Theorem 3.2. Before proving it, we first provide a formal version of the theorem presented in the main text.

Theorem A.1 (Fundamental Connection between SNR and Pass Rate, Formal)**.** *Fix a prompt x. Let $\mathcal{P}(\theta)$ denote the pass rate of x under the current policy $(\pi_\theta(\cdot|x))$: $\mathcal{P}_x(\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[\mathbb{I}(r(y)=1)]$. Consider a random response $y \in \mathcal{Y}$ with binary reward $r(y)$. The advantage of this response is*

$$\mathcal{A}(y) := r(y) - \mathcal{P}_x(\theta).$$

The (stochastic) gradient is thus

$$\widehat{g} := \mathcal{A}(y) \cdot \nabla_\theta \log \pi_\theta(y \mid x).$$

Define the Signal-to-Noise Ratio (SNR) of the stochastic gradient as

$$\mathrm{SNR} := \frac{\|\mathbb{E}\widehat{g}\|^2}{\mathbb{E}\|\widehat{g} - \mathbb{E}\widehat{g}\|^2} = \frac{\|\mathbb{E}\widehat{g}\|^2}{\mathrm{Tr}\big[\mathrm{Cov}[\widehat{g}]\big]}.$$

Then, the SNR satisfies

$$\mathrm{SNR} \le f(\mathcal{P}(\theta)) := \left[\frac{(1-\mathcal{P}_x(\theta))^2}{\mathcal{P}_x(\theta)} + \frac{\mathcal{P}_x(\theta)^2}{1-\mathcal{P}_x(\theta)} - 1\right]^{-1}. \tag{12}$$

Moreover, we have

$$\lim_{\mathcal{P}(\theta)\to 1} \mathrm{SNR} = \lim_{\mathcal{P}(\theta)\to 0} \mathrm{SNR} = 0, \quad \arg\max_{\mathcal{P}(\theta)\in[0,1]} f(\mathcal{P}(\theta)) = \frac{1}{2}.$$

Proof of Theorem A.1. Let π_θ be the LLM and fix a prompt $x \in \mathcal{X}$. For responses $y \in \mathcal{Y} \sim \pi_\theta(\cdot)$, we let $r(y)$ denote the binary reward which will be one if y is correct and zero otherwise. Let $\mathcal{C} \subset \mathcal{Y}$ be the set of correct responses, and $\mathcal{IC} \subset \mathcal{Y}$ be the set of incorrect responses.

Define the pass rate of prompt x, evaluated by the model π_θ, as

$$\mathcal{P}_x(\theta) := \mathbb{P}_{y\sim\pi_\theta(\cdot|x)}\big(r(y)=1\big) = \sum_{y\in\mathcal{C}} \pi_\theta(y \mid x).$$

From the definition of $\widehat{g}$, we know that

$$\mathbb{E}[\widehat{g} \mid r(y)=1] = \sum_{y\in\mathcal{C}} \frac{\pi_\theta(y \mid x)}{\mathcal{P}_x(\theta)} \cdot (1-\mathcal{P}_x(\theta)) \frac{\nabla_\theta \pi_\theta(y \mid x)}{\pi_\theta(y \mid x)} = \frac{1-\mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)} \cdot \nabla_\theta \mathcal{P}_x(\theta),$$

and

$$\mathbb{E}[\widehat{g} \mid r(y)=0] = \sum_{y \in \mathcal{IC}} \frac{\pi_\theta(y \mid x)}{1-\mathcal{P}_x(\theta)} \cdot (-\mathcal{P}_x(\theta)) \frac{\nabla_\theta \pi_\theta(y \mid x)}{\pi_\theta(y \mid x)} = \frac{\mathcal{P}_x(\theta)}{1-\mathcal{P}_x(\theta)} \cdot \nabla_\theta \mathcal{P}_x(\theta).$$

Therefore, one has

$$\mathbb{E}\widehat{g} = \mathcal{P}_x(\theta) \cdot \mathbb{E}[\widehat{g} \mid r(y)=1] + (1-\mathcal{P}_x(\theta)) \cdot \mathbb{E}[\widehat{g} \mid r(y)=0] = \nabla_\theta \mathcal{P}_x(\theta),$$

and

$$\mathbb{E}\left[\mathbb{E}\big[\widehat{g}\big|r(y)\big] \cdot \mathbb{E}\big[\widehat{g}\big|r(y)\big]^\top\right] = \left(\frac{(1-\mathcal{P}_x(\theta))^2}{\mathcal{P}_x(\theta)} + \frac{\mathcal{P}_x(\theta)^2}{1-\mathcal{P}_x(\theta)}\right) \cdot \nabla_\theta \mathcal{P}_x(\theta) \nabla_\theta \mathcal{P}_x(\theta)^\top.$$

Then, we can lower bound the covariance matrix of $\widehat{g}$ by

$$\begin{aligned}\mathrm{Cov}[\widehat{g}] \succeq \mathrm{Cov}\big[\mathbb{E}[\widehat{g} \mid r(y)]\big] &= \mathbb{E}\left[\mathbb{E}\big[\widehat{g}\big|r(y)\big] \cdot \mathbb{E}\big[\widehat{g}\big|r(y)\big]^\top\right] - \mathbb{E}\big[\widehat{g}\big] \cdot \mathbb{E}\big[\widehat{g}\big]^\top \\ &= \left(\frac{(1-\mathcal{P}_x(\theta))^2}{\mathcal{P}_x(\theta)} + \frac{\mathcal{P}_x(\theta)^2}{1-\mathcal{P}_x(\theta)} - 1\right) \cdot \nabla_\theta \mathcal{P}_x(\theta) \nabla_\theta \mathcal{P}_x(\theta)^\top.\end{aligned}$$

Invoking the definition of SNR, we complete the proof □

B. A Fine-grained Analysis on the Signal-to-Noise Ratio

In this section, we provide a fine-grained analysis on the connection between the SNR and the pass rate. We focus on the SNR defined as the (9) with an empirical advantage estimate $\widehat{\mathcal{A}}(y)$ in (8), defined as the difference between the reward of a certain response and the averaged reward of all other responses.

Theorem B.1 (Fundamental Connection between SNR and Pass Rate, Fined-grained Analysis)**.** *Fix a prompt x. Let $\mathcal{P}(\theta)$ denote the pass rate of x under the current policy ($\pi_\theta(\cdot|x)$): $\mathcal{P}_x(\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[\mathbb{I}(r(y)=1)]$. For the prompt x, we generate $y_1, y_2, ..., y_N$ i.i.d. from $\pi_\theta(\cdot \mid x)$ and the gradient estimate is defined as*

$$\widehat{\nabla_\theta J_x}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \widehat{\mathcal{A}}(y_i) \cdot \nabla_\theta \log \pi_\theta(y_i \mid x),$$

where the advantage estimate is from the RLOO estimate:

$$\widehat{\mathcal{A}}(y_i) := r(y_i) - \frac{1}{N-1} \sum_{j \neq i} r(y_j).$$

Then, the SNR of its stochastic gradient estimator (defined in (9)) satisfies

$$\mathrm{SNR} \leq f(\mathcal{P}(\theta)) := \left[\frac{1}{N} \cdot \frac{1}{\mathcal{P}_x(\theta)(1-\mathcal{P}_x(\theta))} + \frac{(N-2)(N-3)}{N(N-1)} - 1\right]^{-1} \tag{13}$$

Moreover, for fixed N, we have

$$\lim_{\mathcal{P}(\theta)\to 1} \mathrm{SNR} = \lim_{\mathcal{P}(\theta)\to 0} \mathrm{SNR} = 0, \quad \arg\max_{\mathcal{P}(\theta)\in[0,1]} f(\mathcal{P}(\theta)) = \frac{1}{2}.$$

Proof of Theorem B.1. Let π_θ be the LLM and fix a prompt $x \in \mathcal{X}$. For responses $y \in \mathcal{Y} \sim \pi_\theta(\cdot)$, we let $r(y)$ denote the binary reward which will be one if y is correct and zero otherwise. Let $\mathcal{C} \subset \mathcal{Y}$ be the set of correct responses, and $\mathcal{IC} \subset \mathcal{Y}$ be the set of incorrect responses.

Define the pass rate of prompt x, evaluated by the model π_θ, as

$$\mathcal{P}_x(\theta) := \mathbb{P}_{y\sim\pi_\theta(\cdot|x)}\big(r(y)=1\big) = \sum_{y\in\mathcal{C}} \pi_\theta(y \mid x).$$

We define the SNR as in Equation (9), where $\widehat{\nabla_\theta J_x}(\theta)$ is defined as

$$\widehat{\nabla_\theta J_x}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \widehat{\mathcal{A}}(y_i)\cdot \nabla_\theta \log \pi_\theta(y_i \mid x), \quad \text{where} \quad y_1, ..., y_N \overset{i.i.d.}{\sim} \pi_\theta(\cdot \mid x).$$

Here, $\widehat{\mathcal{A}}(y_i)$ is the advantage estimate for the response y_i. In RLOO, this is defined as

$$\widehat{\mathcal{A}}(y_i) := r(y_i) - \frac{1}{N-1}\sum_{j\neq i} r(y_j).$$

A simple fact is that the gradient estimate $\widehat{\nabla_\theta J_x}(\theta)$ is unbiased:

$$\begin{aligned}
\mathbb{E}\widehat{\nabla_\theta J_x}(\theta) &= \mathbb{E}\big[\widehat{\mathcal{A}}(y_1)\cdot \nabla_\theta \log \pi_\theta(y_1 \mid x)\big] \\
&= \mathbb{E}\big[r(y_1)\cdot \nabla_\theta \log \pi_\theta(y_1 \mid x)\big] - \frac{1}{N-1}\sum_{i\neq 1}\mathbb{E}\big[r(y_i)\cdot \nabla_\theta \log \pi_\theta(y_1 \mid x)\big] \\
&= \sum_{y_1\in\mathcal{C}} \pi_\theta(y_1 \mid x)\cdot \frac{\nabla_\theta \pi_\theta(y_1 \mid x)}{\pi_\theta(y_1 \mid x)} - \frac{1}{N-1}\sum_{i\neq 1}\mathbb{E}[r(y_i)]\cdot \underbrace{\mathbb{E}[\nabla_\theta \pi_\theta(y_1 \mid x)]}_{=0} \\
&= \nabla_\theta \mathcal{P}_x(\theta).
\end{aligned}$$

We define $R := (r(y_1), r(y_2), r(y_3), ..., r(y_N))^\top \in \{0,1\}^N$ as the random reward vector and $W := \sum_{i=1}^N r(y_i)$ as the empirical pass rate of the prompt x. Now let's consider the covariance matrix of $\widehat{\nabla_\theta J_x}(\theta)$. From the law of total variance, we know

$$\mathrm{Cov}\left[\widehat{\nabla_\theta J_x}(\theta)\right] = \mathrm{Cov}\left[\mathbb{E}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right]\right] + \mathbb{E}\left[\mathrm{Cov}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right]\right] \succeq \mathrm{Cov}\left[\mathbb{E}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right]\right].$$

Here, $A \succeq B$ means $A - B$ is positive semi-definite (PSD) for two PSD matrices A and B. Now we calculate the conditional expectation. We have

$$\begin{aligned}
\mathbb{E}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right] &= \frac{W}{N}\left(1 - \frac{W-1}{N-1}\right)\cdot \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\big[\nabla_\theta \log \pi_\theta(y \mid x)\big| r(y)=1\big] \\
&\quad + \frac{N-W}{N}\left(0 - \frac{W}{N-1}\right)\cdot \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\big[\nabla_\theta \log \pi_\theta(y \mid x)\big| r(y)=0\big] \\
&= \frac{W}{N}\left(1 - \frac{W-1}{N-1}\right)\cdot \sum_{y\in\mathcal{C}} \frac{\pi_\theta(y \mid x)}{\mathcal{P}_x(\theta)}\cdot \frac{\nabla_\theta \pi_\theta(y \mid x)}{\pi_\theta(y \mid x)} \\
&\quad + \frac{N-W}{N}\left(0 - \frac{W}{N-1}\right)\cdot \sum_{y\in\mathcal{IC}} \frac{\pi_\theta(y \mid x)}{1-\mathcal{P}_x(\theta)}\cdot \frac{\nabla_\theta \pi_\theta(y \mid x)}{\pi_\theta(y \mid x)} \\
&= \frac{W(N-W)}{N(N-1)}\cdot \frac{1}{\mathcal{P}_x(\theta)(1-\mathcal{P}_x(\theta))}\cdot \nabla_\theta \mathcal{P}_x(\theta).
\end{aligned}$$

Note that W follows a binomial distribution with parameters N and $\mathcal{P}_x(\theta)$. Recall the moments of binomial random variables, we have

$$\mathbb{E}\left[\mathbb{E}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right]\right] = \mathbb{E}\widehat{\nabla_\theta J_x}(\theta) = \nabla_\theta \mathcal{P}_x(\theta)$$

and

$$\begin{aligned}
&\mathrm{Cov}\left(\mathbb{E}\left[\widehat{\nabla_\theta J_x}(\theta)\middle| R(x)\right]\right) \\
&= \frac{\mathbb{E}\big(W^2(N-W)^2\big)}{N^2(N-1)^2}\cdot \frac{1}{\mathcal{P}_x(\theta)^2(1-\mathcal{P}_x(\theta))^2}\cdot \nabla_\theta \mathcal{P}_x(\theta)\nabla_\theta \mathcal{P}_x(\theta)^\top - \nabla_\theta \mathcal{P}_x(\theta)\nabla_\theta \mathcal{P}_x(\theta)^\top \\
&= \left(\frac{1}{N}\cdot \frac{1}{\mathcal{P}_x(\theta)(1-\mathcal{P}_x(\theta))} + \frac{(N-2)(N-3)}{N(N-1)} - 1\right)\cdot \nabla_\theta \mathcal{P}_x(\theta)\nabla_\theta \mathcal{P}_x(\theta)^\top.
\end{aligned}$$

Therefore, we can upper bound the SNR via

$$\begin{aligned}\text{SNR} &= \frac{\|\mathbb{E}\widehat{\nabla_\theta J_x}(\theta)\|^2}{\mathrm{Tr}\Big[\mathrm{Cov}\big[\widehat{\nabla_\theta J_x}(\theta)\big]\Big]} \le \frac{\|\mathbb{E}\widehat{\nabla_\theta J_x}(\theta)\|^2}{\mathrm{Tr}\Big[\mathrm{Cov}\big[\widehat{\nabla_\theta J_x}(\theta)\big|R(x)\big]\Big]} \\ &= \left[\frac{1}{N}\cdot\frac{1}{\mathcal{P}_x(\theta)(1-\mathcal{P}_x(\theta))} + \frac{(N-2)(N-3)}{N(N-1)} - 1\right]^{-1}.\end{aligned}$$

Invoking the fact that $\mathcal{P}_x(\theta)(1-\mathcal{P}_x(\theta))$ is maximized when $\mathcal{P}_x(\theta) = 1/2$, we complete the proof. □

C. A theoretical justification of SPEED

SPEED filters part of the prompts based on the pass rates estimated via initially generated responses for every prompt, and feeds the current model with precisely moderately difficult prompts. The following theorem shows that SPEED essentially optimizes a transformed objective function, and this objective function is maximized when the pass rate of every prompt is equal to one.

Theorem C.1. *Consider SPEED-RLOO algorithm (Algorithm 1) with $N_{\mathsf{init}} \ge 1, N_{\mathsf{cont}} \ge 1$, and the empirical advantage estimate (8). Assume we sample $x \sim \mathcal{D}_{\mathcal{X}}$ and for every prompt, $y_1, y_2, ..., y_N \overset{i.i.d.}{\sim} \pi_\theta(\cdot \mid x)$ with $N = N_{\mathsf{init}} + N_{\mathsf{cont}}$, and then we use $y_1, ..., y_{N_{\mathsf{init}}}$ to perform the hypothetical generation. Then, the SPEED-RLOO algorithm optimizes the following objective function*

$$\overline{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}}}\big[\Phi(\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)])\big], \tag{14}$$

where

$$\begin{aligned}\Phi(p) = p &- \frac{N_{\mathsf{cont}}}{N(N_{\mathsf{init}}+1)}\Big(p^{N_{\mathsf{init}}+1} - (1-p)^{N_{\mathsf{init}}+1}\Big) \\ &+ \frac{N_{\mathsf{cont}}}{N(N-1)(N_{\mathsf{init}}+1)}\Big(\big(1+N_{\mathsf{init}}p\big)\big(1-p\big)^{N_{\mathsf{init}}} - p^{N_{\mathsf{init}}}\big(N_{\mathsf{init}}\big(1-p\big)+1\big)\Big).\end{aligned}$$

Moreover, the objective function is maximized when $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)] = 1$ for every prompt $x \in \mathcal{X}$.

The theorem show that the SPEED-variants can be viewed as optimizing over a converted objective function. The link function $\Phi(\cdot)$ converts the original pass rate $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)]$ monotonically. Essentially, SPEED converts the optimized objective function via reweighting different propmts according to their pass rates.

Proof of Theorem C.1. Fix N_{init} and N_{cont} with $N = N_{\mathsf{init}} + N_{\mathsf{cont}}$. Let $\overline{J}(\theta)$ denote the equivalent objective function of SPEED-RLOO and $\overline{J}_x(\theta)$ denote the objective function fo prompt x. We have $\overline{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}}}[\overline{J}_x(\theta)]$ by definition and analogously for their gradient. Now we are going to determine the concrete expression of $\nabla_\theta\overline{J}_x(\theta)$. From Algorithm 1, we have

$$\nabla_\theta\overline{J}_x(\theta) = \mathbb{E}_{y_1,y_2,...y_N}\left[\mathbb{I}\Big(\sum_{i=1}^{N_{\mathsf{init}}} r(y_i) \notin \{0, N_{\mathsf{init}}\}\Big)\cdot\frac{1}{N}\sum_{j=1}^{N}\widehat{\mathcal{A}}(y_j)\nabla\log\pi_\theta(y_j \mid x)\right], \tag{15}$$

where the expectation is over $y_1, y_2, ..., y_N \overset{i.i.d.}{\sim} \pi_\theta(\cdot \mid x)$, and $\widehat{\mathcal{A}}(y_j)$ is the advantage estimate for y_j given by

$$\widehat{\mathcal{A}}(y_j) = r(y_j) - \frac{1}{N-1}\sum_{k\neq j} r(y_k).$$

Denote the pass rate to x, evaluated by the model π_θ, as $\mathcal{P}_x(\theta)$. Now we decompose (15) as

$$\begin{aligned}\nabla_\theta \overline{J}_x(\theta) = &\underbrace{\mathbb{E}_{y_1,y_2,\ldots y_N}\left[\widehat{\mathcal{A}}(y_1)\nabla \log \pi_\theta(y_1 \mid x)\right]}_{\text{I}} \\ &\underbrace{- \mathbb{E}_{y_1,y_2,\ldots y_N}\left[\mathbb{I}\Big(\sum_{i=1}^{N_{\text{init}}} r(y_i) = 0\Big) \cdot \frac{1}{N}\sum_{j=1}^{N} \widehat{\mathcal{A}}(y_j)\nabla \log \pi_\theta(y_j \mid x)\right]}_{\text{II}} \\ &\underbrace{- \mathbb{E}_{y_1,y_2,\ldots y_N}\left[\mathbb{I}\Big(\sum_{i=1}^{N_{\text{init}}} r(y_i) = N_{\text{init}}\Big) \cdot \frac{1}{N}\sum_{j=1}^{N} \widehat{\mathcal{A}}(y_j)\nabla \log \pi_\theta(y_j \mid x)\right]}_{\text{III}}.\end{aligned}$$

From the independence among all responses, one has

$$\begin{aligned}\text{I} &= \mathbb{E}_{y_1,y_2,\ldots y_N}\left[r(y_1)\nabla_\theta \log \pi_\theta(y_1 \mid x)\right] - \frac{1}{N-1}\sum_{k\neq 1}\mathbb{E}_{y_1,y_2,\ldots y_N}\left[r(y_k)\nabla_\theta \log \pi_\theta(y_1 \mid x)\right] \\ &= \nabla_\theta \mathcal{P}_x(\theta) - \frac{1}{N-1}\sum_{k\neq 1}\mathbb{E}_{y_1,y_2,\ldots y_N}\big[r(y_k)\big] \cdot \underbrace{\mathbb{E}_{y_1,y_2,\ldots y_N}\big[\nabla_\theta \log \pi_\theta(y_1 \mid x)\big]}_{=0} \\ &= \nabla_\theta \mathcal{P}_x(\theta).\end{aligned}$$

The expectand in II vanishes if $\sum_{i=1}^{N_{\text{init}}} r(y_i) \neq 0$. Analogously, the expectand in III vanished if $\sum_{i=1}^{N_{\text{init}}} r(y_i) \neq N_{\text{init}}$. Let's denote

$$\mathcal{E}_0 := \left\{\sum_{i=1}^{N_{\text{init}}} r(y_i) = 0\right\}, \quad \mathcal{E}_1 := \left\{\sum_{i=1}^{N_{\text{init}}} r(y_i) = N_{\text{init}}\right\}.$$

We have

$$\begin{aligned}&\mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_1 \mid x)\big|\mathcal{E}_0\big] = -\frac{\nabla_\theta \mathcal{P}_x(\theta)}{1-\mathcal{P}_x(\theta)}, \quad \mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_1 \mid x)\big|\mathcal{E}_1\big] = \frac{\nabla_\theta \mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)}, \\ &\mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_N \mid x)\big|\mathcal{E}_0\big] = 0, \quad \mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_N \mid x)\big|\mathcal{E}_1\big] = 0, \\ &\mathbb{E}\big[r(y_N)\nabla_\theta \log \pi_\theta(y_N \mid x)\big|\mathcal{E}_0\big] = \nabla_\theta \mathcal{P}_x(\theta), \quad \mathbb{E}\big[r(y_N)\nabla_\theta \log \pi_\theta(y_N \mid x)\big|\mathcal{E}_1\big] = \nabla_\theta \mathcal{P}_x(\theta).\end{aligned} \tag{16}$$

Therefore, one has

$$\begin{aligned}&\mathbb{E}_{y_1,y_2,\ldots y_N}\left[\mathbb{I}\Big(\sum_{i=1}^{N_{\text{init}}} r(y_i) = 0\Big) \cdot \frac{1}{N}\sum_{j=1}^{N} \widehat{\mathcal{A}}(y_j)\nabla \log \pi_\theta(y_j \mid x)\Big|\mathcal{E}_0\right] \\ &= \mathbb{E}_{y_1,y_2,\ldots y_N}\left[\frac{1}{N}\sum_{j=1}^{N} r(y_j)\left(\nabla_\theta \log \pi_\theta(y_j \mid x) - \frac{1}{N-1}\sum_{k\neq j}\nabla_\theta \log \pi_\theta(y_k \mid x)\right)\Big|\mathcal{E}_0\right] \\ &= \mathbb{E}\left[\frac{1}{N}\sum_{j=N_{\text{init}}+1}^{N} r(y_j)\nabla_\theta \log \pi_\theta(y_j \mid x)\Big|\mathcal{E}_0\right] - \mathbb{E}\left[\frac{1}{N(N-1)}\sum_{j=N_{\text{init}}+1}^{N} r(y_j)\sum_{k\neq j}\nabla_\theta \log \pi_\theta(y_k \mid x)\right)\Big|\mathcal{E}_0\right] \\ &= \frac{N_{\text{cont}}}{N}\nabla_\theta \mathcal{P}_x(\theta) - \frac{N_{\text{cont}}}{N(N-1)}\mathbb{E}\big[r(y_N)\big|\mathcal{E}_0\big] \cdot \mathbb{E}\big[\sum_{k\neq N}\nabla_\theta \log \pi_\theta(y_k \mid x)\big|\mathcal{E}_0\big] \\ &= \frac{N_{\text{cont}}}{N}\nabla_\theta \mathcal{P}_x(\theta) + \frac{N_{\text{cont}}N_{\text{init}}}{N(N-1)}\frac{\mathcal{P}_x(\theta)}{1-\mathcal{P}_x(\theta)} \cdot \nabla_\theta \mathcal{P}_x(\theta).\end{aligned}$$

Therefore, we have

$$\text{II} = \big(1-\mathcal{P}_x(\theta)\big)^{N_{\text{init}}}\left(\frac{N_{\text{cont}}}{N} + \frac{N_{\text{cont}}N_{\text{init}}}{N(N-1)} \cdot \frac{\mathcal{P}_x(\theta)}{1-\mathcal{P}_x(\theta)}\right) \cdot \nabla_\theta \mathcal{P}_x(\theta).$$

Analogously, we have

$$
\begin{aligned}
&\mathbb{E}_{y_1,y_2,\ldots y_N}\left[\mathbb{I}\Big(\sum_{i=1}^{N_{\mathsf{init}}} r(y_i) = N_{\mathsf{init}}\Big)\cdot\frac{1}{N}\sum_{j=1}^{N}\widehat{\mathcal{A}}(y_j)\nabla\log\pi_\theta(y_j\mid x)\Big|\mathcal{E}_1\right]\\
&=\mathbb{E}_{y_1,y_2,\ldots y_N}\left[\frac{1}{N}\sum_{j=1}^{N} r(y_j)\Big(\nabla_\theta\log\pi_\theta(y_j\mid x)-\frac{1}{N-1}\sum_{k\neq j}\nabla_\theta\log\pi_\theta(y_k\mid x)\Big)\Big|\mathcal{E}_1\right]\\
&=\frac{N_{\mathsf{init}}}{N}\mathbb{E}\big[\nabla_\theta\log\pi_\theta(y_1\mid x)\big|\mathcal{E}_1\big]-\frac{N_{\mathsf{init}}}{N(N-1)}\sum_{k\neq 1}\mathbb{E}\big[\nabla_\theta\log\pi_\theta(y_k\mid x)\big|\mathcal{E}_1\big]\\
&\qquad+\frac{N_{\mathsf{cont}}}{N}\mathbb{E}\big[r(y_N)\nabla_\theta\log\pi_\theta(y_N\mid x)\big|\mathcal{E}_1\big]-\frac{N_{\mathsf{cont}}}{N(N-1)}\sum_{k\neq N}\mathbb{E}\big[r(y_N)\nabla_\theta\log\pi_\theta(y_k\mid x)\big|\mathcal{E}_1\big]\\
&=\left(\frac{N_{\mathsf{init}}}{N}-\frac{N_{\mathsf{init}}(N_{\mathsf{init}}-1)}{N(N-1)}\right)\cdot\frac{\nabla\mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)}\\
&\qquad+\frac{N_{\mathsf{cont}}}{N}\nabla_\theta\mathcal{P}_x(\theta)-\frac{N_{\mathsf{cont}}}{N(N-1)}\mathcal{P}_x(\theta)\cdot\sum_{k\neq N}\mathbb{E}\big[\nabla_\theta\log\pi_\theta(y_N\mid x)\big|\mathcal{E}_1\big]\\
&=\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\cdot\frac{\nabla_\theta\mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)}+\frac{N_{\mathsf{cont}}}{N}\nabla_\theta\mathcal{P}_x(\theta)-\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\cdot\nabla_\theta\mathcal{P}_x(\theta)\\
&=\frac{N_{\mathsf{cont}}}{N}\nabla_\theta\mathcal{P}_x(\theta)+\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\cdot\frac{1-\mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)}\cdot\nabla_\theta\mathcal{P}_x(\theta).
\end{aligned}
$$

Therefore, one has

$$
\text{III}=\mathcal{P}_x(\theta)^{N_{\mathsf{init}}}\left(\frac{N_{\mathsf{cont}}}{N}+\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\frac{1-\mathcal{P}_x(\theta)}{\mathcal{P}_x(\theta)}\right)\cdot\nabla_\theta\mathcal{P}_x(\theta).
$$

This indicates

$$
\begin{aligned}
\nabla_\theta\overline{J}_x(\theta)=\Bigg[&1-\frac{N_{\mathsf{cont}}}{N}\Big(\mathcal{P}_x(\theta)^{N_{\mathsf{init}}}+\big(1-\mathcal{P}_x(\theta)\big)^{N_{\mathsf{init}}}\Big)\\
&-\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\Big(\mathcal{P}_x(\theta)\big(1-\mathcal{P}_x(\theta)\big)^{N_{\mathsf{init}}-1}+\big(1-\mathcal{P}_x(\theta)\big)\mathcal{P}_x(\theta)^{N_{\mathsf{init}}-1}\Big)\Bigg]\nabla_\theta\mathcal{P}_x(\theta).
\end{aligned}
$$

Integrating the gradient gives:

$$
\overline{J}_x(\theta)=\Phi(\mathcal{P}_x(\theta)),
$$

where

$$
\begin{aligned}
\Phi(p)=p&-\frac{N_{\mathsf{cont}}}{N(N_{\mathsf{init}}+1)}\Big(p^{N_{\mathsf{init}}+1}-(1-p)^{N_{\mathsf{init}}+1}\Big)\\
&+\frac{N_{\mathsf{cont}}}{N(N-1)(N_{\mathsf{init}}+1)}\Big(\big(1+N_{\mathsf{init}}p\big)\big(1-p\big)^{N_{\mathsf{init}}}-p^{N_{\mathsf{init}}}\big(N_{\mathsf{init}}\big(1-p\big)+1\big)\Big)+\text{Const.}
\end{aligned}
$$

Moreover, since

$$
\begin{aligned}
\Phi'(p)&=1-\frac{N_{\mathsf{cont}}}{N}\big(p^{N_{\mathsf{init}}}+(1-p)^{N_{\mathsf{init}}}\big)-\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\big(p(1-p)^{N_{\mathsf{init}}-1}+(1-p)p^{N_{\mathsf{init}}-1}\big)\\
&\geq 1-\frac{N_{\mathsf{cont}}}{N}-\frac{N_{\mathsf{init}}N_{\mathsf{cont}}}{N(N-1)}\\
&\geq 1-\frac{N_{\mathsf{cont}}}{N}-\frac{N_{\mathsf{init}}}{N}=0. && (N_{\mathsf{cont}}\leq N-1)
\end{aligned}
$$

Therefore, $\Phi(\cdot)$ is monotonically increasing and hence, for every prompt $x\in\mathcal{X}$, one has

$$
\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)]=1\quad\text{maximizes}\quad\Phi(\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(y)]).
$$

This completes the proof. □

Algorithm 2 Selective Prompting with Efficient Estimation of Difficulty (SPEED)

Input: generative model π_θ; binary reward model r; difficulty thresholds $P_{\text{low}}, P_{\text{high}}$; generation counts $N_{\text{init}}, N_{\text{cont}}$; total training steps T; batch size B
Initialize buffer $\mathcal{D}_{\text{buffer}} \leftarrow \emptyset$, cache $\mathcal{D}_{\text{accepted}} \leftarrow \emptyset$, step counter $t \leftarrow 0$
repeat
  if $|\mathcal{D}_{\text{buffer}}| < B$ **then**
    Fetch a new batch of prompts $\mathcal{D}_{\text{new}}$ from the data loader
    $\mathcal{D}_{\text{infer}} \leftarrow \mathcal{D}_{\text{new}} \cup \mathcal{D}_{\text{accepted}}$
    Generate N_{init} responses for each $x \in \mathcal{D}_{\text{new}}$ and N_{cont} responses for each $x \in \mathcal{D}_{\text{accepted}}$
    $\mathcal{D}_{\text{buffer}} \leftarrow \mathcal{D}_{\text{buffer}} \cup \mathcal{D}_{\text{accepted}}$
    for all $x \in \mathcal{D}_{\text{new}}$ **do**
      $\textsc{PassRate}(x) \leftarrow \dfrac{1}{N_{\text{init}}} \sum_{i=1}^{N_{\text{init}}} \mathbb{I}\big(\text{response}_i \text{ correct}\big)$
    end for
    $\mathcal{D}_{\text{accepted}} \leftarrow \mathcal{D}_{\text{accepted}} \cup \{x \in \mathcal{D}_{\text{new}} \mid P_{\text{low}} < \textsc{PassRate}(x) < P_{\text{high}}\}$
  else
    Sample $\mathcal{D}_{\text{train}} \subset \mathcal{D}_{\text{buffer}}$ with $|\mathcal{D}_{\text{train}}| = B$
    Perform one RL update step on $\mathcal{D}_{\text{train}}$
    $\mathcal{D}_{\text{buffer}} \leftarrow \mathcal{D}_{\text{buffer}} \setminus \mathcal{D}_{\text{train}}$
    $t \leftarrow t + 1$
  end if
until $t = T$
Output: trained model π_θ

D. Full Algorithm

Now we describe our full algorithm combined with the data buffer and the pre-fetching mechanism.

E. Figures of More Experimental Results

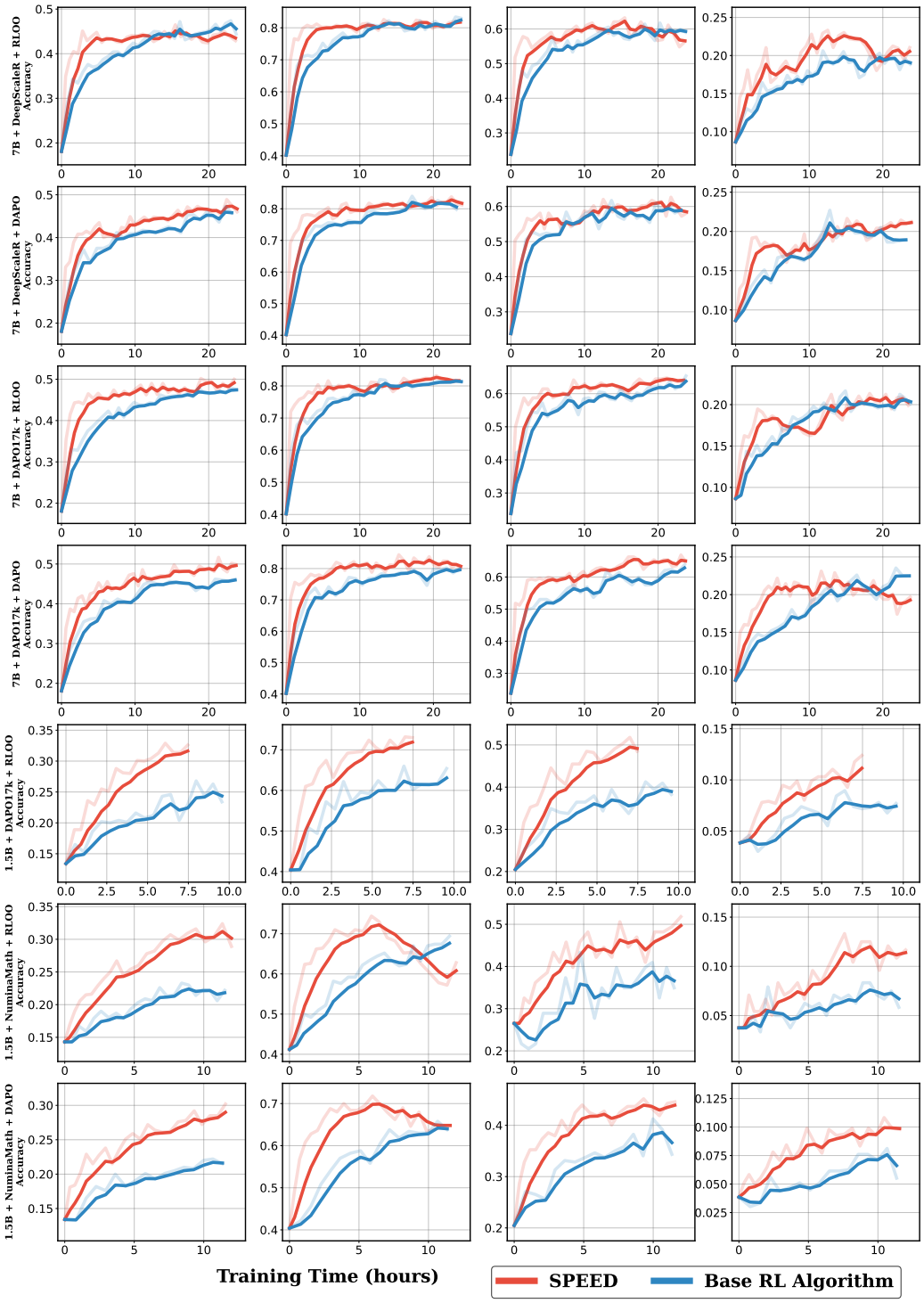

Figure 6: Validation accuracy on various mathematical reasoning benchmarks for SPEED-variants of RL algorithms, and base RL algorithms. The initial model used is Qwen2.5-Math-7B (for the top 4 rows) and Qwen2.5-Math-1.5B (for the bottom 3 rows). Y-axis labels follow the pattern 'Model size + Training set + Base RL algorithm'. We use three training dataset: NuminaMath (Li et al., 2024), DAPO-17k (without 1k held-out validation set) (Yu et al., 2025), and DeepScaleR (Luo et al., 2025), and we use 2 base RL algorithms: RLOO (Ahmadian et al., 2024) and DAPO (Yu et al., 2025). The lighter curves represent raw accuracy results, while the bold curves indicate smoothed results obtained via an exponential moving average.