TRAINING REASONING MODELS ON SATURATED PROBLEMS VIA FAILURE-PREFIX CONDITIONING

Anonymous authors
Paper under double-blind review

ABSTRACT

As Reinforcement Learning with Verifiable Rewards (RLVR) substantially improves the reasoning abilities of large language models (LLMs), a new bottleneck emerges: more training problems become saturated, that is, the LLM answers the questions correctly for nearly every rollout. On such problems, rewards provide little useful learning signal. While collecting harder problems is a natural response, it is costly and increasingly difficult. We propose *failure-prefix conditioning*, a simple method that unlocks the remaining signal in saturated problems by shifting exploration toward failure-prone reasoning states. By conditioning on prefixes of rare incorrect trajectories, the method improves the model's ability to recover from misleading early reasoning. We observe that failure-prefix conditioning consistently improves performance where standard RLVR stalls, and achieves gains comparable to training on newly collected medium-difficulty problems. We further analyze the model's robustness, finding that our method reduces performance degradation under misleading failure prefixes, albeit with a mild trade-off in adherence to correct early reasoning. Finally, we demonstrate that an iterative approach, which refreshes failure prefixes during training, unlocks additional gains after performance plateaus. Overall, our results show that saturated problems still contain valuable learning signal, and that failure-prefix conditioning provides an effective way to unlock it. [1]

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has substantially improved the reasoning capabilities of large language models (LLMs) (OpenAI, 2024; Guo et al., 2025; Lambert et al., 2025). Today's frontier models achieve near-ceiling scores on advanced benchmarks such as MATH and AIME (Akhtar et al., 2026) and attain gold-medal-level performance on the IMO, IOI and ICPC (El-Kishky et al., 2025; Google DeepMind, 2025; Luong et al., 2025). Beyond such standardized evaluations, they are now producing results beyond established human knowledge, contributing to improved matrix multiplication algorithms (Novikov et al., 2025), solutions to multiple Erdős problems (Alexeev et al., 2026; Feng et al., 2026), a counterexample to the Jacobian conjecture (Alpöge, 2026), and a proposed resolution of the Navier–Stokes existence and smoothness problem (OpenAI, 2026).

However, this progress also creates a new bottleneck. As models improve, more training problems become saturated, that is, the LLM answers the questions correctly for nearly every rollout. On such problems, rewards provide little useful learning signal (Yu et al., 2025; Kim et al., 2025; Zeng et al., 2026). Continuing progress would then seem to require a supply of harder, unsaturated problems, but manually collecting such problems only becomes more difficult and more expensive over time (Villalobos et al., 2024).

Instead of collecting ever more challenging training data, we address *how to extract as much learning signal as possible from the saturated training problems in the existing dataset*. Importantly, saturation does not imply that the learning signal is exhausted. Even on problems with near-perfect rollout accuracy, incorrect reasoning trajectories still exist, albeit extremely rarely under standard sampling. As established in prior work, such informative failures play a crucial role in training (Zhu et al.,

[1] Code available at `https://anonymous.4open.science/r/failure-prefix-conditioning-0760`

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

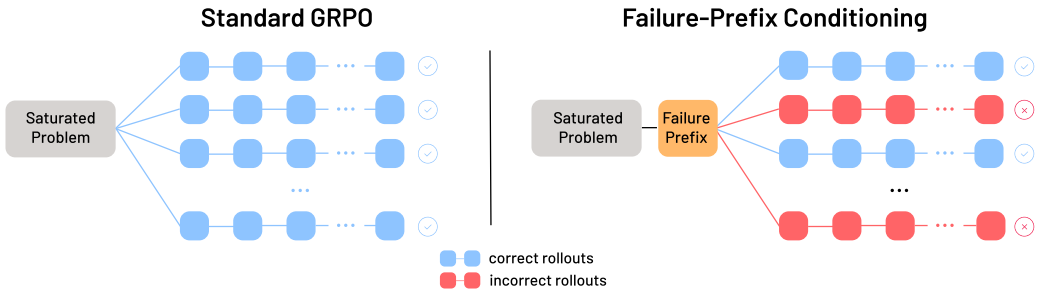

Figure 1: Illustration of standard GRPO training and training with failure-prefix conditioning (ours). While standard GRPO predominantly generates correct rollouts, failure-prefix conditioning exposes the model to failure-prone reasoning states, making informative failures more accessible.

2025b; Setlur et al., 2026b). However, most of the sampling budget is spent generating redundant correct solutions, with failures becoming exceedingly sparse (Wang et al., 2025). This suggests that the core challenge in learning from saturated problems is not the *absence of informative failures*, but their *poor accessibility*. If failures could be encountered more frequently, RLVR could potentially continue to make progress even on saturated problems.

To this end, we propose *failure-prefix conditioning*, a simple and effective method for learning from saturated problems in RLVR. As shown in Figure 1, instead of sampling responses from the original question, this method explicitly targets failure-prone reasoning states. We identify rare incorrect rollouts from saturated questions, and create new prompts consisting of the original question followed by a prefix of the incorrect rollout. We then select prefix lengths such that the resulting rollout accuracy is close to a target value τ, which controls the difficulty of the resulting training examples. In practice, we set $\tau = 0.5$, where the learning signal is maximized (Li et al., 2025). Training on the resulting dataset directs exploration toward failure-prone regions of the response space and improves its ability to recover from misleading early reasoning.

We evaluate our method in a high-accuracy training regime, where many training questions are already saturated. This setting reflects the setting faced by today's frontier reasoning models. Specifically, we use DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) as the base model and the MATH train set (Hendrycks et al., 2021) as the training data. We identify saturated questions, construct the failure-prefix-conditioned dataset, and train the model with RLVR. We compare against several baselines designed to isolate the effect of our method: standard RLVR on the full MATH training set, standard RLVR on the same saturated subset without conditioning, and standard RLVR on an equal number of medium-difficulty questions. These comparisons test whether failure-prefix conditioning extracts more learning signals from saturated problems and whether it can match the benefit of collecting new high-signal problems.

Across math reasoning benchmarks of varying difficulty, training on the full MATH set or directly on the saturated subset provides negligible improvement, confirming that standard RLVR stalls in this regime. In contrast, failure-prefix conditioning yields consistent gains over the base model, demonstrating that our method can effectively recover learning signal from saturated problems. We further show this improved performance is on par with that of instead training on an equal number of additional medium-difficulty questions. We also find that failure-prefix conditioning is robust to the exact target accuracy. Although $\tau = 0.5$ performs best, nearby choices still yield strong gains. Motivated by this robustness, we consider a lower-cost *fixed-truncation* variant: instead of estimating rollout accuracy for all candidate prefixes and selecting the one closest to τ, it simply truncates each incorrect trajectory at a fixed fraction γ of its length. This achieves comparable performance while substantially reducing the overhead in dataset construction.

Next, we evaluate how well our method improves the model's ability to recover from misleading reasoning. As compared with baseline models, models trained with failure-prefix-conditioned datasets are substantially better at recovering from misleading reasoning. When given incorrect intermediate steps, their performances degrade more slowly, showing stronger robustness to early mistakes. We also observe a mild trade-off: when given correct intermediate steps, these models show a smaller accuracy increase than the baselines, showing tendency to diverge more from correct early reasoning. Overall, however, the robustness gains dominate, leading to improved aggregate performance.

Finally, we explore whether learning from saturated problems can be further extended by iteratively refreshing failure prefixes as the model improves. As training progresses, performance under a fixed failure-prefix-conditioned dataset eventually plateaus, suggesting that previously informative prefixes may become less effective as the policy updates. Therefore, we perform a second iteration of failure-prefix conditioning, resampling new failures from the updated model and reconstructing the prefix-conditioned dataset. Empirically, this iterative procedure yields additional gains beyond the initial plateau, suggesting that periodically updating failure prefixes may further recover additional learning signal and enable continued improvement.

We summarize our contributions as follows:

- We propose failure-prefix conditioning, a method that directs exploration toward failure-prone reasoning states to enable effective RLVR training on saturated problems.
- We show that failure-prefix conditioning enhances robustness to incorrect intermediate reasoning trajectories, enabling better recovery from misleading partial solutions compared to standard RLVR baselines.
- We demonstrate that iteratively refreshing failure prefixes can yield additional gains after performance plateaus, offering a potential pathway for further extending learning on saturated problems.

Overall, our results establish that saturated problems still contain valuable learning signal, and that failure-prefix conditioning is an effective method for unlocking their potential.

2 Preliminaries: RLVR and Saturated Problems

In this section, we explain why training with GRPO (Shao et al., 2024), a standard RLVR algorithm, stalls on saturated problems. Section 2.1 reviews the training procedure and notation, and Section 2.2 analyzes why saturation leads to stalled learning.

2.1 GRPO Training

Let $\mathcal{D} = \{(q, a^*)\}$ denote a dataset of questions q with verifiable ground-truth answers a^*. Given a policy π_θ, we generate N independent rollouts

$$y_i \sim \pi_\theta(\cdot \mid q), \quad i = 1, \ldots, N,$$

and assign each rollout a binary reward $r_i = r(y_i; q)$, where $r_i = 1$ if y_i provides the correct answer a^* and $r_i = 0$ otherwise. For each rollout in $\{y_i\}_{i=1}^N$, the advantage is given by $A_i = (r_i - \mu_r)/(\sigma_r + \epsilon)$, where $\mu_r = \frac{1}{N}\sum_{i=1}^N r_i$, $\sigma_r = \mathrm{std}(\{r_i\}_{i=1}^N)$, and $\epsilon > 0$ is a small constant that prevents division by zero when $\sigma_r = 0$.

The policy π_θ is then updated by minimizing the following loss, where we omit the clipping and KL-regularization terms for clarity:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{(q,a^*)\sim\mathcal{D}}\left[\frac{1}{N}\sum_{i=1}^{N} A_i \log \pi_\theta(y_i \mid q)\right].$$

2.2 Training with Saturated Problems

We say that a problem q is *saturated* under policy π_θ if the rollout accuracy $p_\theta(q)$ is close to 1, where

$$p_\theta(q) := \Pr_{y\sim\pi_\theta(\cdot|q)}[r(y; q) = 1]$$

For such a problem, with high probability all sampled rollouts satisfy $r_i = 1$, yielding

$$\mu_r = 1, \quad \sigma_r = 0,$$

and consequently $A_i = 0$ for all i. As a result, the policy gradient vanishes and training stalls.

Even when a rare incorrect rollout occurs, reward variance remains small. For binary rewards,

$$\mathrm{std}[r] = \sqrt{p_\theta(q)(1 - p_\theta(q))},$$

which approaches zero as $p_\theta(q) \to 1$. Since the policy gradient magnitude scales with this standard deviation value (Li et al., 2025) (see Appendix I for a full derivation), learning signals on saturated problems are weak. This analysis is also empirically supported by the findings of Zhan et al. (2025), who observe substantially smaller gains from training on easy questions ($p_\theta(q) \in [0.75, 1.00)$) compared to moderately difficult ones ($p_\theta(q) \in (0.25, 0.75]$).

3 METHODOLOGY

We propose *failure-prefix conditioning*, a simple and effective method for learning from saturated problems in RLVR. The core idea is to reshape exploration by conditioning training on failure-prone reasoning states, instead of initiating exploration from the original question. Section 3.1 provides the motivation for our approach, and Section 3.2 presents the proposed methodology.

3.1 MOTIVATION

As analyzed in Section 2.2, learning stalls on saturated problems not because the model is perfect, but because its failures are too rare under standard RLVR. Even when rollout accuracy is close to 1, incorrect responses still exist in the model's response space. However, during training, these failure-prone states are encountered too rarely to provide meaningful learning signal. In other words, the fundamental limitation is not the *absence of informative failures*, but their *poor accessibility*.

To address this, rather than repeatedly sampling from the question as in the standard setting, we target exploration toward regions of the response space where uncertainty is higher and failures occur more frequently. Concretely, we propose *failure-prefix conditioning*: condition training on prefixes obtained from rare incorrect responses, enabling RLVR to begin exploration directly from failure-prone states and improving the model's ability to recover from misleading early reasoning.

This simple change dramatically improves accessibility. Consider a saturated problem with rollout accuracy $p_\theta(q) \approx 0.95$. Under standard sampling, failures occur with probability 0.05, meaning only a small number of incorrect responses are expected, and most rollouts are redundant successes that contribute little learning signal. If, instead, training begins from a failure-prone state where, say, $p \approx 0.50$, the same sampling budget yields substantially more incorrect responses. As a result, informative failures become far more accessible.

3.2 METHODOLOGY: FAILURE-PREFIX CONDITIONING

Let π_θ be a fixed policy and $\mathcal{D}_{\text{saturated}}$ be a set of saturated questions, defined as questions whose rollout accuracy $p_\theta(q)$ is close to 1. For each such question q, we obtain at least one incorrect rollout $\tilde{y}$ produced by π_θ. From each incorrect rollout $\tilde{y}$, we then construct a set of prefixes

$$\mathcal{S}(q) = \{\tilde{y}_{1:\alpha_k}\}_{k=1}^{K},$$

where $\tilde{y}_{1:\alpha_k}$ denotes the first α_k tokens of $\tilde{y}$. In our experiments in Section 4, we sweep through prefix lengths corresponding to $10\%, 20\%, \ldots, 90\%$ of the total trajectory length, yielding $K = 9$ candidate prefixes per question.

Each prefix $s \in \mathcal{S}(q)$ defines a prefix-conditioned prompt $q \oplus s$, from which we estimate the prefix-conditioned rollout accuracy

$$p_\theta(q, s) = \Pr_{y \sim \pi_\theta(\cdot|q \oplus s)}[r(y; q) = 1]$$

using a fixed number of rollouts. Intuitively, when generation starts from the original question q, the rollout accuracy is near one for saturated problems. As progressively longer prefixes of an incorrect trajectory are prefixed, the conditioning context increasingly constrains the model toward failure states, and the rollout accuracy tends to decay towards 0 (Wen et al., 2025). While this decay is not necessarily monotonic, sweeping over prefix lengths reliably yields intermediate prefixes whose rollout accuracy lies between these extremes.

We select a single prefix $s^{(q)}$ whose rollout accuracy is closest to a target value $\tau \in (0, 1)$:

$$s^{(q)} = \arg\min_{s \in \mathcal{S}(q)} \left| p_\theta(q, s) - \tau \right|.$$

In our main experiments, we set $\tau = 0.5$, where binary rewards exhibit the highest variance and GRPO provides the strongest learning signal (Li et al., 2025).

Using the selected prefixes, we construct a new training set

$$\mathcal{D}' = \{(q \oplus s^{(q)}, a^*) \mid q \in \mathcal{D}_{\text{saturated}}\}.$$

GRPO training is then performed on $\mathcal{D}'$ following the standard procedure described in Section 2.1.

4 Main Experiment

4.1 Experiment Settings

Model and dataset. We consider a setting where the base model already achieves high accuracy on the training data, with many questions already saturated. This mirrors the regime faced by today's frontier reasoning models. Specifically, we use DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) as the base model and the MATH (Hendrycks et al., 2021) train set of 7.5k questions as the training data. We estimate question-level rollout accuracy by generating 128 rollouts per question. The base model obtains 82.6% accuracy on the training set, with the distribution shown in Appendix D.

Failure-prefix conditioning. We divide the questions into 3 groups: (1) *128/128*: fully saturated questions for which no incorrect rollout is observed; (2) *121–127/128*: saturated questions with rollout accuracy greater than 120/128 and at least one observed incorrect rollout; and (3) *0–120/128*: unsaturated questions with rollout accuracy of $\leq$120/128. These groups contain 2,455, 2,147, and 2,898 questions, respectively. We sample one incorrect rollout from each question in the 121–127/128 group and apply failure-prefix conditioning following Algorithm 1 with target accuracy $\tau = 0.5$, where reward variance is maximized. We also consider $\tau = 0.25$ and $\tau = 0.75$ for ablation studies. The cutoff of $120/128$ $(= 15/16)$ is chosen to match our RLVR training configuration, where each question is sampled 16 times. Since an accuracy of $15/16$ is the largest value at which one expects at least one incorrect rollout in 16 independent samples, we treat questions above this level as saturated. Additional inference details are provided in Appendix C.

Baselines. We train the base model on failure-prefix-conditioned datasets with different τ values. We compare against the six baselines, each designed to examine a specific factor.

1. *Base*: The base model.
2. *Full MATH*: Standard RLVR on the full MATH train set, which tests whether our method improves over conventional training on all available data.
3. *Saturated*: Standard RLVR on the 121–127/128 questions, isolating the effect of failure-prefix conditioning versus standard training on the same data.
4. *Medium*: Standard RLVR on the same number of 2,147 medium-difficulty questions sampled from DeepScaleR (Tan et al., 2025) (a harder dataset than MATH), selected to have rollout accuracy 14–18/32 ($\approx 50\%$) under the base model, where binary reward variance maximizes. This evaluates whether failure-prefix conditioning can match or exceed the benefit of collecting an equal number of new high-learning-signal questions.
5. *Failure-Prefix + Unsaturated*: Failure-prefix-conditioned data ($\tau = 0.5$) combined with 0–120/128 questions.
6. *Unsaturated*: Standard RLVR on the 0–120/128 questions alone, isolating the contribution of unsaturated data. These last two baselines examine whether our method provides complementary gains when paired with unsaturated problems in the train set.

RLVR training. We train the base model using Dr.GRPO (Liu et al., 2025c), a variant of GRPO that mitigates response length bias. All models use the same configuration with 16 rollouts per question, representing a standard, non-aggressive exploration setting. Further details are provided in Appendix B.

Evaluation. We evaluate all models on five math reasoning benchmarks spanning a wide range of difficulties (from easiest to hardest): MATH500 (Hendrycks et al., 2021), AMC12 (AI-MO, 2025), AIME24 (math-ai, 2025a), AIME25 (math-ai, 2025b), and HMMT25 (MathArena, 2025). To

Table 1: Performance across math and other-domain benchmarks. Numbers in parentheses indicate absolute improvements over the base model. The two average columns report unweighted averages within their respective groups. Best results are shown in **bold**, and second-best results are underlined.

	Math						Other Domains			
Model	**MATH 500**	**AMC 12**	**AIME 24**	**AIME 25**	**HMMT 25**	**Avg.**	**MBPP+**	**GPQA-D**	**BBH-LD**	**Avg.**
Base	83.5	53.1	28.4	24.0	14.2	40.6	55.5	31.2	52.3	46.3
Full MATH	84.5 (+1.0)	54.6 (+1.5)	27.8 (-0.6)	24.2 (+0.2)	13.6 (-0.6)	40.9 (+0.3)	54.1 (-1.4)	31.9 (+0.7)	55.6 (+3.3)	47.2 (+0.9)
Saturated	84.3 (+0.8)	52.7 (-0.4)	29.0 (+0.6)	24.0 (+0.0)	13.5 (-0.7)	40.7 (+0.1)	55.2 (-0.3)	31.0 (-0.2)	54.6 (+2.3)	46.9 (+0.6)
Medium	85.7 (+2.2)	58.8 (+5.7)	34.2 (+5.8)	26.2 (+2.2)	15.0 (+0.8)	44.0 (+3.4)	56.8 (+1.3)	33.2 (+2.0)	66.9 (+14.6)	52.3 (+6.0)
Failure-Prefix + Unsaturated	85.7 (+2.2)	55.6 (+2.5)	30.7 (+2.3)	25.8 (+1.8)	13.8 (-0.4)	42.3 (+1.7)	55.4 (-0.1)	32.5 (+1.3)	56.8 (+4.5)	48.2 (+1.9)
Unsaturated	86.0 (+2.5)	56.5 (+3.4)	26.0 (-2.4)	25.7 (+1.7)	14.2 (+0.0)	41.7 (+1.1)	54.7 (-0.8)	32.0 (+0.8)	58.7 (+6.4)	48.5 (+2.2)
Failure-Prefix ($\tau = 0.5$)	**86.6** (+3.1)	**60.5** (+7.4)	33.1 (+4.7)	27.1 (+3.1)	**15.4** (+1.2)	**44.5** (+3.9)	57.1 (+1.6)	33.7 (+2.5)	**67.1** (+14.8)	**52.6** (+6.3)
Failure-Prefix ($\tau = 0.25$)	85.6 (+2.1)	58.2 (+5.1)	**35.5** (+7.1)	**27.7** (+3.7)	14.2 (+0.0)	44.2 (+3.6)	56.3 (+0.8)	**34.1** (+2.9)	66.0 (+13.7)	52.1 (+5.8)
Failure-Prefix ($\tau = 0.75$)	85.6 (+2.1)	57.8 (+4.7)	35.0 (+6.6)	26.2 (+2.2)	15.1 (+0.9)	43.9 (+3.3)	56.7 (+1.2)	33.4 (+2.2)	65.4 (+13.1)	51.8 (+5.5)
Fixed Truncation ($\gamma = 0.25$)	86.4 (+2.9)	59.3 (+6.2)	32.8 (+4.4)	26.4 (+2.4)	14.6 (+0.4)	43.9 (+3.3)	**57.3** (+1.8)	32.8 (+1.6)	65.6 (+13.3)	51.9 (+5.6)
Fixed Truncation ($\gamma = 0.5$)	86.4 (+2.9)	58.5 (+5.4)	34.3 (+5.9)	26.9 (+2.9)	15.3 (+1.1)	44.3 (+3.7)	56.8 (+1.3)	33.4 (+2.2)	65.9 (+13.6)	52.0 (+5.7)
Fixed Truncation ($\gamma = 0.75$)	85.7 (+2.2)	58.3 (+5.2)	35.3 (+6.9)	26.6 (+2.6)	14.5 (+0.3)	44.1 (+3.5)	56.9 (+1.4)	33.1 (+1.9)	64.9 (+12.6)	51.6 (+5.3)

assess generalization beyond mathematics, we additionally evaluate on MBPP+ (Liu et al., 2023), GPQA-Diamond (Rein et al., 2023), and the Logical Deduction task from BIG-Bench Hard (BBH-LD) (Suzgun et al., 2023), representing code generation, scientific reasoning, and logical deduction, respectively. For each question, we generate 32 samples and report the mean accuracy to ensure statistical robustness. Details of the evaluation setup are provided in Appendix C.

4.2 Results

Failure-prefix conditioning enables learning from saturated problems. Table 1 shows that our method consistently improves performance across all evaluated math benchmarks, from the easiest (MATH500) to the most challenging (HMMT25). The average math accuracy rises from 40.6% for the base model to 44.5%, corresponding to a +3.9 point gain. Furthermore, this improvement extends beyond mathematics, with the average accuracy on the other-domain benchmarks increasing from 46.3% to 52.6% (+6.3). In contrast, standard RLVR yields little improvement on either benchmark group when trained on the full MATH training set (40.9% on math; 47.2% on other domains) or on the same saturated questions without conditioning (40.7% on math; 46.9% on other domains), consistent with the saturation analysis in Section 2.2. These results show that failure-prefix conditioning recovers useful learning signal from saturated problems that standard RLVR cannot effectively exploit.

Failure-prefix conditioning matches the benefit of new medium-difficulty data. Medium-difficulty problems are effective for RLVR because its rollout accuracy is close to 50%, where learning signal is maximized. Nevertheless, our method slightly outperforms training on medium-difficulty questions on both math (44.5% vs. 44.0%) and other domains (52.6% vs. 52.3%), despite training only with the original saturated MATH questions. This indicates that failure-prefix conditioning can extract a learning signal from saturated questions that is comparable to, or even stronger than, training on newly collected unsaturated medium-difficulty problems. Additional experiments addressing response-length issue during training are provided in Appendix F.

Failure-prefix conditioning is robust to the target accuracy. We set the main target accuracy to $\tau = 0.5$ since this theoretically maximizes reward variance and thus the expected gradient signal, as discussed in Section 3.2. To test sensitivity to this hyperparameter, we repeat training with $\tau = 0.25$ and $\tau = 0.75$. As shown in Table 1, both variants remain effective: $\tau = 0.25$ reaches 44.2% (+3.6), and $\tau = 0.75$ reaches 43.9% (+3.3), only slightly below $\tau = 0.5$ at 44.5%. The same pattern holds across the other-domain benchmarks. This implies that, while targeting maximal variance performs best, our method does not require precise tuning of the target accuracy. This result motivates a lower-cost variant of our method in Section 4.3.

Combining unsaturated data with failure-prefix conditioning is not always beneficial. In practical RLVR training pipelines, it is natural to use all available unsaturated problems. However, as shown in Table 1, combining failure-prefix-conditioned data with the 0–120/128 unsaturated subset yields average accuracies of only 42.3% on math and 48.2% on other domain. Although both remain above the base-model averages, they are substantially lower than failure-prefix conditioning alone (44.5%

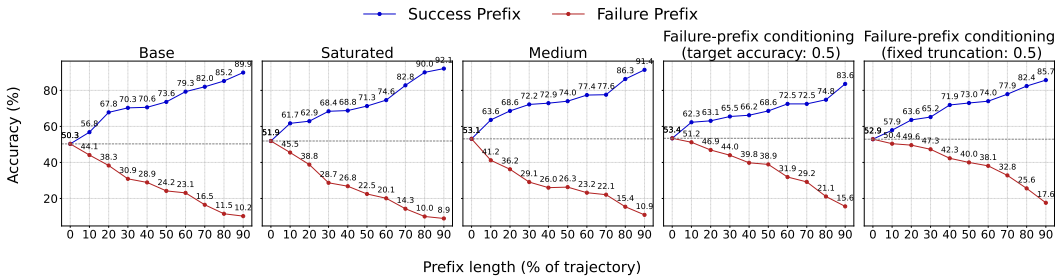

Figure 2: Rollout accuracy versus prefix length (% of trajectory) when conditioning on correct and incorrect prefixes.

and 52.6%, respectively). Meanwhile, training on the unsaturated subset alone reaches 41.7% on math and 48.5% on other-domain benchmarks, indicating that these examples are not harmful by themselves. Rather, the result suggests that simple data mixing can dilute the benefit of failure-prefix conditioning. A possible reason is that many unsaturated examples may quickly become saturated during training and contribute weaker signals later on, though further study is needed. Given this, we conduct the remaining analyses using the failure-prefix-conditioned dataset alone.

4.3 Reducing Prefix Selection Overhead via Fixed Truncation

A major source of computational overhead in our method comes from prefix length selection. Matching a target accuracy requires slicing each incorrect response at multiple lengths and estimating rollout accuracy for each candidate prefix, which is expensive (Appendix E). However, Section 4.2 shows that performance is insensitive to the exact target value τ, motivating a cheaper alternative.

We try *fixed truncation*, where each incorrect response is truncated at a fixed fraction γ of its length without estimating prefix-conditioned accuracy. We test $\gamma \in \{0.25, 0.5, 0.75\}$ following Algorithm 2. As shown in Table 1, these variants achieve math averages of 43.9%, 44.3%, and 44.1%, and other-domain averages of 51.9%, 52.0%, and 51.6%, respectively. These results are close to explicit $\tau = 0.5$ matching, which achieves 44.5% on math and 52.6% on the other-domain benchmarks. This small gap suggests that precise prefix-difficulty calibration is unnecessary and that fixed truncation offers a practical variant that significantly reduces overhead while preserving most of the benefits.

5 Recovery From Misleading Reasoning

Failure-prefix conditioning directs exploration toward failure-prone regions of the response space, enabling the model to better recover from early mistakes. In this section, we evaluate how well our method improves recovery from misleading early reasoning.

Recall that, in Section 4.2, we evaluate each model by generating 32 responses for every MATH500 question. We reuse the same rollouts to analyze recovery behavior. We compare five models: *Base*, *Saturated*, *Medium*, and two variants of our method: (i) *FP-target*, which selects prefixes to match target accuracy $\tau = 0.5$, and (ii) *FP-fixed*, which uses a fixed truncation ratio $\gamma = 0.5$.

We select the subset of questions for which all models produce at least one correct and one incorrect response, yielding 131 questions. This ensures that recovery is feasible for all models. For each model and question, we sample one on-policy incorrect response and construct prefixes at 10%, 20%, ..., 90% of its length. We use model-specific prefixes because our goal is to measure each model's ability to recover from its own errors at inference time. Conditioning on each prefix, we generate 32 continued rollouts and compute rollout accuracy (details in Appendix C). We group and average rollout accuracies by the percentage values and analyze how accuracy degrades as the prefix extends further into an incorrect reasoning trajectory.

5.1 Failure-prefix conditioning improves robustness to misleading prefixes

As shown in Figure 2, all models exhibit declining accuracy as the prefix length increases, consistently showing the "tunnel vision" issue in LLM reasoning (Wen et al., 2025). However, FP-target and FP-fixed degrade substantially more slowly. At a 30% prefix, FP-target drops by 6.5 points (53.4 to 46.9) and FP-fixed by 3.3 points (52.9 to 49.6), compared to 12.0 points for the base model (50.3 to 38.3), 13.1 points for the saturated model (51.9 to 38.8) and 16.9 points for the medium model. This gap persists across prefix lengths, indicating stronger robustness to misleading intermediate states.

Notably, the medium model—despite having similar overall accuracy—shows significantly sharper degradation than the base and saturated models. This suggests that improved robustness is specific to failure-prefix conditioning, rather than a byproduct of higher overall performance.

5.2 A mild trade-off on correct prefixes

Additionally, we examine whether failure-prefix conditioning induces excessive deviation from correct reasoning when given an initially correct partial solution, which is an undesired behavior. To assess this, we also perform an identical experiment but with prefixes from a correct response.

Figure 2 shows that the FP models benefit less from correct prefixes. For example, at 30% prefix, FP-target improves by 12.1 points (53.4 to 65.5) and FP-fixed by 12.3 points (52.9 to 65.2), compared to 20.0 points (50.3 to 70.3) for the base, 16.5 points (51.9 to 68.4) for the saturated, and 19.1 points for the medium model (53.1 to 72.2). This pattern holds across the entire range of prefix lengths.

Overall, these results reveal a mild trade-off: failure-prefix conditioning substantially improves robustness to misleading reasoning, while slightly reducing gains from correct prefixes. Addressing this behavior may require further investigation in future work. Importantly, this effect is limited in magnitude and does not outweigh the robustness gains achieved under failure-prefix conditioning, as reflected in the overall performance improvements reported in Section 4.2.

6 Iterative Failure-Prefix Conditioning

We observe that performance eventually plateaus as training with failure-prefix conditioning ($\tau = 0.5$) progresses, with the mean reward flattening (Figure 3, left). This behavior may arise from diminishing learning signal in the conditioned dataset, or from fixed failure prefixes becoming increasingly off-policy as the model improves.

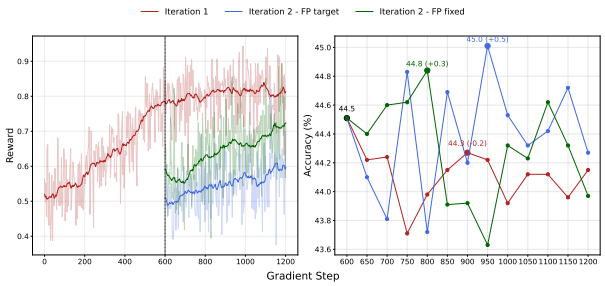

Figure 3: Training dynamics of iterative failure-prefix conditioning. *Left:* Training reward curves for prolonged iteration 1 (steps 0–1200) and iteration 2 with updated failure prefixes through step 1200. *Right:* Mean accuracy across math benchmarks for both iterations measured from steps 600-1200, with peak performance point for each model highlighted.

To test whether additional gains can be obtained, we perform a second iteration of failure-prefix conditioning. Starting from the best checkpoint at step 600, we resample 128 responses for each of the MATH training questions. From these, we identify 2,124 questions with rollout accuracy 121–127/128. For each, we construct two failure-prefix-conditioned datasets: *FP-target*, which selects prefixes to match a target accuracy $\tau = 0.5$, and *FP-fixed*, which uses a fixed truncation ratio $\gamma = 0.5$. We then continue training from the checkpoint for an additional 600 gradient steps on each dataset.

We find that this second iteration yields further improvements. As shown in Figure 3 (right), extending training in the first iteration does not surpass the peak accuracy of 44.5 achieved at step 600. In contrast, the second iteration improves performance to 45.0 (+0.5) for FP-target and 44.8 (+0.3) for FP-fixed over the original checkpoint. A similar pattern is observed on the other-domain benchmarks (Appendix G). These results point to the possibility that iteratively refreshing failure prefixes may help recover additional learning signal and enable continued improvement.

7 Related Work

Task Difficulty for RL Training of LLMs. Prior work shows that policy updates in RL training of LLMs depend strongly on task difficulty, with learning signals maximized when task success rates are at an intermediate level (Li et al., 2025; Razin et al., 2024). Zhan et al. (2025) empirically supports this by stratifying tasks by difficulty and showing that gains peak on moderate-difficulty examples. Building on this insight, several methods regulate task difficulty during RL training. DAPO filters prompts with extremely low or high rollout accuracy, which contribute little to parameter updates (Yu et al., 2025). Other approaches order fixed training tasks from easier to harder to improve training efficiency (Chen et al., 2025; Du et al., 2025; Shi et al., 2025). RLVE instead adapts the training environment to control task difficulty relative to the model (Zeng et al., 2026). Self-play methods similarly let LLMs generate tasks at suitable difficulty levels (Liu et al., 2025a; Zhao et al., 2025). Our work shares the goal of maintaining effective difficulty, but differs by focusing on saturated problems: rather than filtering, reordering, or generating tasks, we recover their utility by conditioning training on failure-prone states from partial incorrect trajectories.

Curriculum learning and context-enhanced learning for RL training of LLMs. Curriculum learning has been widely studied as a way to regulate task difficulty and improve training efficiency (Baker et al., 2020; Jiang et al., 2020; Portelas et al., 2020), and has recently been extended to RL training for large language models (Chen et al., 2025; Gao et al., 2025). In the context of LLMs, many such approaches operate by augmenting the input context with additional information that facilitates reasoning during training (Xi et al., 2024). Some studied methods provide partial ground-truth hints in the context for difficult problems where the base model fails to produce successful rollouts, enabling the model to generate correct trajectories during training and yielding improvements that transfer effectively to settings without such hints (Zhang et al., 2025; Qu et al., 2026; Setlur et al., 2026a). Similarly, Zhu et al. (2025a) inject additional information into the context—excluded from autoregressive gradient computation—to achieve exponentially improved sample efficiency on multi-step reasoning tasks. Our approach also modifies the input context during RL training, but essentially inverts this paradigm: whereas prior methods inject *correct* information to assist with *difficult* problems, we condition on *incorrect* trajectories to increase the difficulty of *easy* (saturated) problems. This exposes failure-prone states and allows the recovery of learning signals that are otherwise poorly accessible.

Scaling RL training for performance boosts. Scaling RL training has repeatedly been shown to improve the reasoning performance of LLMs (Ettinger et al., 2025; Guo et al., 2025). ProRL demonstrates that substantially increasing the number of gradient updates can improve both pass@1 and pass@k performance (Liu et al., 2025b). BroRL further extends this line of work by dramatically increasing per-problem exploration, sampling hundreds of rollouts per prompt to surpass the saturation point observed in ProRL and achieve additional gains (Hu et al., 2025). Relatedly, Wang et al. (2025) show that intensive training on a single problem can extract sufficient learning signal to match the performance gains obtained from training on datasets with over a thousand examples. Taken together, these results indicate that scaling RL training through increased gradient steps or exploration can drive further performance gains. However, it incurs prohibitive computational costs and memory overhead, with the vast majority of the sampling budget being consumed by generating redundant correct solutions during training. In contrast, our approach extracts learning signals from saturated problems efficiently, reallocating exploration to failure-prone states rather than simply increasing the sampling budget.

8 Conclusion

In this paper, we introduce failure-prefix conditioning, a simple and effective method that extends RLVR training on saturated problems where standard methods stall. By explicitly initializing training from rare incorrect partial trajectories, our approach recovers critical learning signals, yielding performance gains comparable to training on medium-difficulty tasks while enhancing robustness to misleading reasoning. Moreover, we demonstrate that iteratively refreshing these prefixes enables sustained improvement beyond initial plateaus. Ultimately, our work establishes that saturated problems remain a valuable resource for training reasoning models, provided exploration is directed toward their sparse but informative failure modes.

LIMITATIONS AND FUTURE WORK

While our work suggests that saturated problems can still be useful for RLVR with failure-prefix conditioning, it has several limitations that motivate future work.

First, failure-prefix conditioning requires collecting rare incorrect rollouts from saturated problems, which introduces considerable computational overhead (Appendix E). While existing work has largely focused on improving the probability of correct generation, an important future direction is to develop efficient methods for sampling rare incorrect on-policy trajectories from saturated problems.

Second, our experiments use a relatively small model, for which unsaturated training data remains accessible. However, failure-prefix conditioning is most relevant for frontier models, where acquiring new unsaturated problems becomes significantly more difficult. Further work is needed to determine whether the same benefits hold at larger scales.

Third, as discussed in Section 5, failure-prefix conditioning improves robustness to misleading early reasoning steps, but may mildly reduce adherence to correct intermediate reasoning. Future work is needed to preserve recovery ability while maintaining consistency with correct reasoning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with editing the manuscript for clarity, presenting and summarizing experimental results, implementing portions of the training and evaluation code, and formatting LaTeX tables and references. We also used these tools to help identify relevant literature. Generative AI was not used to generate or modify experimental data or results, develop the core research idea or theoretical framework, formulate mathematical claims or proofs, or independently design or conduct experiments. All AI-assisted code was reviewed and tested by the authors, and all AI-assisted text, citations, calculations, and tables were manually verified against the original sources and experimental outputs. We take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the collection of sensitive data. Our experiments use established research benchmarks and model-generated outputs. Our work is intended to advance the scientific understanding of reinforcement learning for LLM reasoning. We have reported our methodology and experimental results transparently and take responsibility for the research practices and conclusions presented in this paper.

REPRODUCIBILITY STATEMENT

We provide a detailed description of our method and evaluation protocol in the main text, with exact training hyperparameters, implementation details, and additional experimental settings documented in the appendix. The code and data required to reproduce our experiments are available through the anonymous repository linked in the supplementary materials. Together, these resources specify the data construction, training, and evaluation procedures used to obtain the reported results.

REFERENCES

AI-MO. Amc12 2022, 2023 dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc, 2025. Hugging Face dataset.

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When ai benchmarks plateau: A systematic study of benchmark saturation, 2026. arXiv:2602.16763.

Boris Alexeev, Moe Putterman, Mehtaab Sawhney, Mark Sellke, and Gregory Valiant. Short proofs in combinatorics, probability and number theory ii, 2026. arXiv:2604.06609.

Levent Alpöge. Hello there the jacobian conjecture is false. Post on X, July 2026. URL https://x.com/__alpoge__/status/2079028340955197566. Announcement of an explicit counterexample in dimension three; credits Anthropic's Fable model for work leading to the example.

Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *Proceedings of the International Conference on Learning Representations (ICLR 2020)*, 2020.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv:2505.14970*, 2025.

Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv:2501.12599*, 2025.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, et al. Competitive programming with large reasoning models, 2025. arXiv:2502.06807.

Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. *arXiv:2512.13961*, 2025.

Tony Feng, Trieu Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang-hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, Carlo Pagano, Yuri Chervonyi, Dawsen Hwang, Kaiying Hou, Sergei Gukov, Cheng-Chiang Tsai, Hyunwoo Choi, Youngbeom Jin, Wei-Yuan Li, Hao-An Wu, Ruey-An Shiu, Yu-Sheng Shih, Quoc V. Le, and Thang Luong. Semi-autonomous mathematics discovery with gemini: A case study on the erdős problems, 2026. arXiv:2601.22401.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training. *arXiv:2510.01135*, 2025.

Google DeepMind. Gemini achieves gold-medal level at the international collegiate programming contest world finals. https://deepmind.google/blog/gemini-achieves-gold-medal-level-at-the-international-collegiate-programming-contest-world-finals/, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

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. In *Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*, 2021.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, and Yi Dong. Brorl: Scaling reinforcement learning via broadened exploration. *arXiv:2510.01180*, 2025.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, 2020.

Minwu Kim, Anubhav Shrestha, Safal Shrestha, Aadim Nepal, and Keith Ross. Reinforcement learning vs. distillation: Understanding accuracy and capability in llm reasoning. *arXiv:2505.14216*, 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 ACM Symposium on Operating Systems Principles (SOSP '23)*, pp. 611–626, Koblenz, Germany, 2023. Association for Computing Machinery. doi: 10.1145/3600006.3613165.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Proceedings of the Conference on Language Modeling (COLM 2025)*, 2025.

Gang Li, Ming C. Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. Disco: Reinforcing large reasoning models with discriminative constrained optimization. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. arXiv:2505.12366.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *Proceedings of the International Conference on Learning Representations 2026 (ICLR 2026)*, 2025a. arXiv:2506.24119.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Proceedings of the Thirty-Seventh Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025b. arXiv:2505.24864.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Proceedings of the 2025 Conference on Language Modeling (COLM 2025)*, 2025c.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP 2025)*, 2025.

math-ai. Aime24 dataset. https://huggingface.co/datasets/math-ai/aime24, 2025a. Hugging Face dataset.

math-ai. Aime25 dataset. https://huggingface.co/datasets/math-ai/aime25, 2025b. Hugging Face dataset.

MathArena. Hmmt25 dataset (february 2025). https://huggingface.co/datasets/MathArena/hmmt_feb_2025, 2025. Hugging Face dataset.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. arXiv:2506.13131.

OpenAI. Openai o1 system card, 2024. arXiv:2412.16720.

OpenAI. Finite time blowup for navier–stokes. https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf, September 2026.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 2020.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv:2601.18779*, 2026.

Noam Razin, Hattie Zhou, Omid Saremi, Vimal Thilak, Arwen Bradley, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. Vanishing gradients in reinforcement finetuning of language models. In *Proceedings of International Conference on Learning Representations (ICLR 2024)*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark, 2023. arXiv:2311.12022.

Amrith Setlur, Zijian Wang, Andrew Cohen, Paria Rashidinejad, and Sang Michael Xie. Reuse your flops: Scaling rl on hard problems by conditioning on very off-policy prefixes, 2026a. arXiv:2601.18795.

Amrith Setlur, Matthew Y. R. Yang, Charlie Victor Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. In *Proceedings of the International Conference on Learning Representations 2026 (ICLR 2026)*, 2026b. arXiv:2506.09026.

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

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. URL https://aclanthology.org/2023.findings-acl.824/.

Sijun Tan, Michael Luo, Justin Wong, Colin Cai, Xiaoxiang Shi, William Yuan Tang, Manan Roongta, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Effective rl scaling of reasoning models via iterative context lengthening. In *ICLR 2026 Conference Submission*, 2025. URL https://openreview.net/forum?id=I6GzDCne7U.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? limits of llm scaling based on human-generated data, 2024. arXiv:2211.04325.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. arXiv:2504.20571.

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. 2025. arXiv:2509.04475.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. arXiv:2503.14476.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, Chenyang Zhao, Yulia Tsvetkov, Simon Shaolei Du, Natasha Jaques, Hao Peng, Pang Wei Koh, and Hannaneh Hajishirzi. Rlve: Scaling up reinforcement learning for language models with adaptive verifiable environments. In *Proceedings of the 43rd International Conference on Machine Learning 2026 (ICML 2026)*, 2026. arXiv:2511.07317.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. In *Proceedings of the International Conference on Learning Representations 2026 (ICLR 2026)*, 2025. arXiv:2510.02245.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. arXiv:2506.17211.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. arXiv:2505.03335.

Xingyu Zhu, Abhishek Panigrahi, and Sanjeev Arora. On the power of context-enhanced learning in llms. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, 2025a. arXiv:2503.01821.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. In *Proceedings of the Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025b. arXiv:2506.01347.

A FAILURE-PREFIX CONDITIONING ALGORITHMS

Algorithm 1 describes the full failure-prefix conditioning procedure used in our main experiments. For each saturated question, we first obtain an incorrect rollout and construct multiple candidate prefixes by truncating it at different lengths. We then estimate the rollout accuracy induced by each prefix and select the one whose accuracy is closest to the target accuracy τ. This produces a prefix-conditioned dataset that shifts exploration toward failure-prone states with high reward variance.

Algorithm 1 Failure-Prefix Conditioning with Target Accuracy

Require: Saturated dataset $\mathcal{D}_{\text{saturated}} = \{(q, a^*)\}$, policy π_θ, target accuracy τ, number of rollouts N
Ensure: Prefix-conditioned dataset $\mathcal{D}'$

Initialize $\mathcal{D}' \leftarrow \emptyset$
for each $(q, a^*) \in \mathcal{D}_{\text{saturated}}$ **do**
  Obtain an incorrect rollout $\tilde{y} \sim \pi_\theta(\cdot \mid q)$
  Construct candidate prefix set

$$\mathcal{S}(q) = \{\tilde{y}_{1:\alpha_k}\}_{k=1}^{K}$$

  for each prefix $s \in \mathcal{S}(q)$ **do**
    Sample N rollouts $\{y_i\}_{i=1}^{N} \sim \pi_\theta(\cdot \mid q \oplus s)$
    Estimate prefix-conditioned rollout accuracy

$$p_\theta(q, s) \leftarrow \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[r(y_i; q) = 1]$$

  end for
  Select prefix

$$s^{(q)} \leftarrow \arg\min_{s \in \mathcal{S}(q)} |p_\theta(q, s) - \tau|$$

  Add $(q \oplus s^{(q)}, a^*)$ to $\mathcal{D}'$
end for
return $\mathcal{D}'$

Algorithm 2 gives the lower-cost fixed-truncation variant. Instead of estimating rollout accuracy for multiple candidate prefixes, it keeps a fixed fraction γ of each incorrect rollout. This avoids the prefix-selection overhead while preserving most of the benefit of failure-prefix conditioning.

Algorithm 2 Failure-Prefix Conditioning with Fixed Truncation

Require: Saturated dataset $\mathcal{D}_{\text{saturated}} = \{(q, a^*)\}$, policy π_θ, truncation fraction $\gamma \in (0, 1)$
Ensure: Prefix-conditioned dataset $\mathcal{D}'_\gamma$

Initialize $\mathcal{D}' \leftarrow \emptyset$
for each $(q, a^*) \in \mathcal{D}_{\text{saturated}}$ **do**
  Obtain an incorrect rollout $\tilde{y} \sim \pi_\theta(\cdot \mid q)$
  Let $L \leftarrow |\tilde{y}|$
  Set prefix length $\alpha \leftarrow \lfloor \gamma L \rfloor$
  Construct fixed-truncation prefix $s^{(q)} \leftarrow \tilde{y}_{1:\alpha}$
  Add $(q \oplus s^{(q)}, a^*)$ to $\mathcal{D}'$
end for
return $\mathcal{D}'$

B GRPO TRAINING DETAILS

Library For GRPO training in this work, we use the `GRPOTrainer` from the `TRL`[2] library

Prompt Setting We use the Qwen-Math chat template for all GRPO training and evaluation. When applying failure-prefix conditioning, the prefix is inserted immediately after the `<think>` token, as shown below.

Qwen-Math template

```
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.
<|im_end|>
<|im_start|>user
{question}
<|im_end|>
<|im_start|>assistant
<think>{prefix}
```

Reward Function We adopt a minimalistic reward setting. A response received a reward of 1 if it contained the correct final answer, and 0 otherwise. Answer verification is performed using the `math_verify`[3] package, consistent with the one used for evaluation.

$$R(q, a^*, y) = \begin{cases} 1 & \text{if the response } y \text{ to question } q \text{ matches the ground truth answer } a^* \\ 0 & \text{otherwise} \end{cases}$$

Clip-Higher Following DAPO (Yu et al., 2025), we adopt the clip-higher strategy to mitigate entropy collapse during training. Specifically, we set $\epsilon_{\text{high}} = 0.4$ and $\epsilon_{\text{low}} = 0.2$, with number of iterations per batch as 2.

Training Hyperparameters Table 2 summarizes the key hyperparameters used in RLVR training. Note that, with this configuration, running approximately 1,200 gradient steps takes about 5 days on 4 A100 80GB GPUs, corresponding to roughly 480 GPU-hours.

Table 2: Key hyperparameters used for RLVR training.

Hyperparameter	Value
Optimizer	AdamW
Learning rate scheduler	Constant
Maximum response length	8000
Temperature	1.0
Top-p	1.0
Number of rollouts per question	16
Number of iterations per batch	2
ϵ_{high}	0.4
ϵ_{low}	0.2
Global batch size	4 (per device) $\times$ 4 (GPUs) $\times$ 10 (accumulation) = 160
Learning rate	1×10^{-6}
Tensor parallel size	2
KL divergence weight	0

[2] https://github.com/huggingface/trl
[3] https://github.com/huggingface/Math-Verify

C Inference Details

Libraries For inference, we use `vLLM` (Kwon et al., 2023) for response generation. For response grading, we use the `math_verify` package, consistent with the one used for rewarding during RLVR training.

Prompt Setting For all evaluation and rollout accuracy measurement experiments, we use the same prompt setting described in Appendix B.

Inference Hyperparameters for Measuring Rollout Accuracy To measure rollout accuracy for identifying saturated questions and determining the best prefix length, we run inference from each prefix-conditioned question using the same setting as in generating rollouts during GRPO training (Appendix B). Specifically, we use the configuration shown in Table 3. Note that, when identifying saturated questions, if the single incorrect rollout exceeds the maximum generation length of 8000 tokens, we exclude the corresponding question from the dataset, since responses truncated at the token limit cannot be reliably classified as correct or incorrect. In practice, this exclusion affects very few questions: problems with rollout accuracy $\geq 120/128$ are generally not difficult, and almost all generated responses terminate well within the 8000-token limit.

Table 3: Key hyperparameters used for inference for identifying saturated questions and determining best prefix length.

Hyperparameter	**Value**
Maximum token length	8000
Temperature	1.0
Top-p	1.0
Number of rollouts per question	32

Inference Hyperparameters for Evaluation Table 4 summarizes the key hyperparameters used in inference for evaluation in Section 4.2, Section 4.3 and Section 6. The same setting is also used for measuring the recovery from failure prefixes in Section 5.

Table 4: Key hyperparameters used for inference during evaluation.

Hyperparameter	**Value**
Maximum token length	32000
Temperature	0.6
Top-p	1.0
Number of rollouts per question	32

D TRAINING DATA DETAILS

MATH Train Set. Figure 4 shows the distribution of question-level rollout accuracy for the base model on the MATH training set.

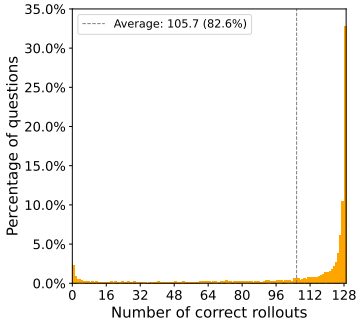

Figure 4: Question-level rollout accuracy of the base model on the MATH training set.

Failure-Prefix-Conditioned Dataset We construct failure-prefix-conditioned datasets using two selection strategies: target accuracy and fixed truncation. Figure 5 shows the distributions of selected prefix lengths and their resulting rollout accuracies under different target values τ, covering both the main experiments in Section 4.2 and iterative failure-prefix conditioning in Section 6. Figure 6 reports the corresponding rollout-accuracy distributions for fixed truncation with different γ values.

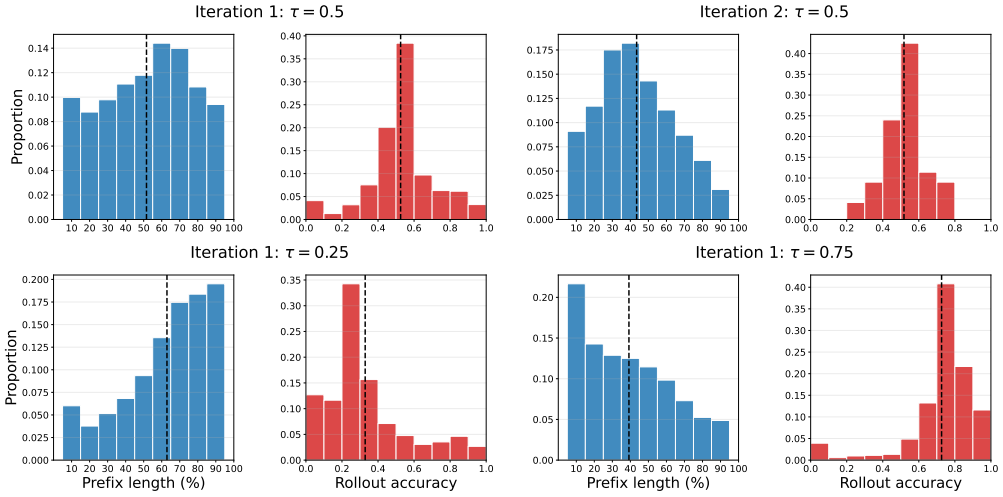

Figure 5: Failure-prefix-conditioned datasets selected by target accuracy. Each pair of plots shows the distribution of selected prefix lengths and the resulting prefix-conditioned rollout accuracies. Dashed lines indicate mean values.

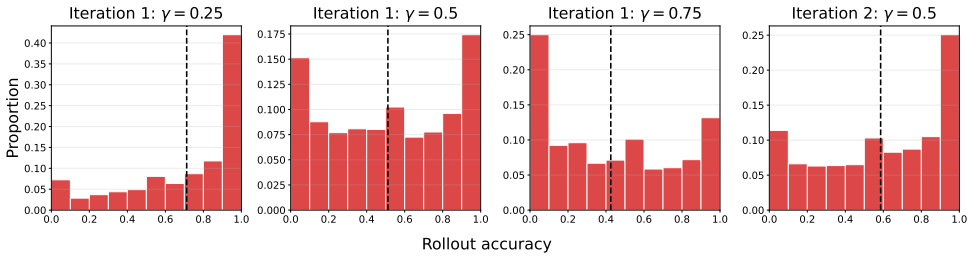

Figure 6: Prefix-conditioned rollout accuracy distributions for fixed-truncation datasets with different γ values. Dashed lines indicate mean values.

E Computation Overhead for Failure-Prefix Conditioning

Table 5: Token overhead of failure-prefix conditioning and simple cost-reduction strategies. For saturation identification, staged sampling reduces the cost from 128 rollouts per question to a selective procedure that continues sampling only for initially saturated questions. For prefix selection, using a coarser 20% prefix grid and 8 rollouts per prefix substantially reduces the estimation cost.

Stage	Full cost	Reduced cost	Relative cost
Saturation identification	3.32B tokens	0.64B tokens	19.3%
Prefix selection	1.94B tokens	0.11B tokens	5.8%

Failure-prefix conditioning introduces additional computation in two stages. First, we sample multiple responses per training question to estimate rollout accuracy and identify saturated questions with rare failures. Second, for each candidate failure prefix, we estimate prefix-conditioned rollout accuracy to select the prefix closest to the target accuracy.

Table 5 reports the token usage of each stage. The initial saturation-identification step uses 3.32B tokens when we sample 128 rollouts for every question. The prefix-selection step uses 1.94B tokens when we evaluate candidate prefixes on the full 10%, 20%, . . ., 90% grid with 32 rollouts per prefix.

These costs can be reduced with simple approximations. For saturation identification, instead of sampling 128 rollouts for every question, we first sample 16 rollouts and only continue sampling for questions with 16/16 correct responses, stopping once an incorrect rollout is observed or the maximum of 128 attempts is reached. This staged procedure reduces the cost from 3.32B to 0.64B tokens, corresponding to 19.3% of the original budget.

For prefix selection, we can use a coarser prefix grid and fewer rollouts per prefix. Specifically, replacing the 10% grid with 20% increments and reducing the number of rollouts from 32 to 8 lowers the cost from 1.94B to 0.11B tokens, or 5.8% of the original budget. These results suggest that the computational overhead of failure-prefix conditioning can be substantially reduced without changing the overall training objective.

F Additional Experiment: Addressing Response Length Issue

Table 6: Performance across math and other-domain benchmarks with controls for response length during RLVR training. The two average columns report unweighted averages within their respective groups.

	Math						Other Domains			
Model / Setting	**MATH 500**	**AMC 12**	**AIME 24**	**AIME 25**	**HMMT 25**	**Avg.**	**MBPP+**	**GPQA-D**	**BBH-LD**	**Avg.**
Failure-Prefix, uncapped budget	86.6	60.5	33.1	27.1	15.4	44.5	57.1	33.7	67.1	52.6
Failure-Prefix, prefix-counted budget	86.2	59.1	34.7	26.5	14.8	44.3	56.4	32.6	67.5	52.2
Medium data, 8k limit	85.7	58.8	34.2	26.2	15.0	44.0	56.8	33.2	66.9	52.3
Medium data, 12k limit	86.2	59.5	35.1	25.8	15.2	44.4	57.2	33.1	67.4	52.7

All RLVR experiments in the paper use a response length limit of 8,000 tokens, as explained in Appendix B. However, this raises two potential fairness concerns.

First, failure-prefix-conditioned examples receive a larger effective budget because generation begins after the prefix, while the prefix itself is not counted. To address this, we rerun failure-prefix conditioning with $\tau = 0.5$, counting the prefix length toward the 8,000-token limit and excluding the small number of prefixes that already exceed this limit. As shown in Table 6, the math average remains close to the original result (44.3% vs. 44.5%), as does the other-domain average (52.2% vs. 52.6%). This suggests that the improvement comes from failure-prefix conditioning itself rather than from a larger effective token budget.

Second, standard RLVR on medium-difficulty questions may be disadvantaged by the same 8,000-token limit, since these questions produce longer responses on average and are more likely to be truncated. Increasing its response limit to 12,000 tokens improves the math average from 44.0% to 44.4% and the other-domain average from 52.3% to 52.6%. Although length therefore matters for this baseline, its performance remains comparable to failure-prefix conditioning. Our conclusion is thus unchanged: applying failure-prefix conditioning to saturated problems is competitive with collecting new high-signal data.

G Full Results for Iterative Failure-Prefix Conditioning

Table 7: Benchmark-level performance after the second iteration of failure-prefix conditioning.

	Math						Other Domains			
Model / Setting	**MATH 500**	**AMC 12**	**AIME 24**	**AIME 25**	**HMMT 25**	**Avg.**	**MBPP+**	**GPQA-D**	**BBH-LD**	**Avg.**
Iteration 2 – Target Accuracy ($\tau = 0.5$)	86.7	60.2	35.7	27.1	15.2	45.0	56.9	33.5	68.6	53.0
Iteration 2 – Fixed Truncation ($\gamma = 0.5$)	86.4	59.9	34.9	27.4	15.4	44.8	57.1	33.2	67.3	52.6

Table 7 reports the benchmark-level results for the second iteration of failure-prefix conditioning described in Section 6. Both variants improve over the first-iteration checkpoint, with the target-accuracy variant reaching the best average performance.

H Success-Prefix Conditioning Experiment

Table 8: Comparison with success-prefix conditioning. The two average columns report unweighted averages within their respective benchmark groups.

	Math						Other Domains			
Model / Setting	**MATH 500**	**AMC 12**	**AIME 24**	**AIME 25**	**HMMT 25**	**Avg.**	**MBPP+**	**GPQA-D**	**BBH-LD**	**Avg.**
Base	83.5	53.1	28.4	24.0	14.2	40.6	55.5	31.2	52.3	46.3
Full MATH	84.5	54.6	27.8	24.2	13.6	40.9	54.1	31.9	55.6	47.2
Saturated	84.3	52.7	29.0	24.0	13.5	40.7	55.2	31.0	54.6	46.9
Success-Prefix ($\gamma = 0.5$)	84.1	53.2	28.0	23.8	13.8	40.6	54.3	30.9	53.3	46.2
Failure-Prefix ($\tau = 0.5$)	86.6	60.5	33.1	27.1	15.4	44.5	57.1	33.7	67.1	52.6

Methods such as PrefixRL and POPE use correct prefixes to make difficult problems easier and thereby move their success rates toward an intermediate regime with stronger learning signal (Qu et al., 2026; Setlur et al., 2026a). Our setting requires the opposite intervention: because saturated problems already have near-perfect rollout accuracy, correct prefixes should make them even easier and further reduce access to informative failures.

To distinguish the effect of failure prefixes from that of prefix conditioning itself, we evaluate a success-prefix baseline. For each saturated training problem, we sample a correct response, truncate it at a fixed ratio of $\gamma = 0.5$, and apply the same RLVR training procedure to the resulting prefix-conditioned dataset. Table 8 shows that success-prefix conditioning yields little improvement over the base model, reaching averages of 40.6% on math and 46.2% on the other-domain benchmarks. Its performance is also similar to standard RLVR on the full MATH dataset and the saturated subset, while remaining substantially below failure-prefix conditioning. These results indicate that the gains arise specifically from conditioning on failure prefixes rather than from prefix conditioning alone.

I Derivation of Question-Level Weights and Their Relation to Reward Variance

We provide a detailed derivation showing how question-level weights arise in GRPO-style objectives and why they are directly tied to the standard deviation of the reward distribution, following the analysis of Li et al. (2025).

Setup We consider a binary reward setting where, for a given question q, each generated output o receives a reward

$$r(o \mid q) \in \{0, 1\}.$$

Let

$$p(q) := \mathbb{E}_{o \sim \pi_{\text{old}}(\cdot|q)}[r(o \mid q)]$$

denote the probability that the current policy π_{old} produces a correct answer for question q.

Under this setting, the reward variance and standard deviation satisfy

$$\text{Var}[r(o \mid q)] = p(q)\big(1 - p(q)\big), \qquad \text{std}[r(o \mid q)] = \sqrt{p(q)\big(1 - p(q)\big)}.$$

Normalized advantage GRPO and its variants employ a normalized advantage function of the form

$$A(o \mid q) = \frac{r(o \mid q) - p(q)}{\sqrt{p(q)(1 - p(q))}}.$$

Since the reward is binary, this admits the explicit piecewise representation

$$A(o \mid q) = \begin{cases} \sqrt{\dfrac{1 - p(q)}{p(q)}}, & \text{if } r(o \mid q) = 1, \\ -\sqrt{\dfrac{p(q)}{1 - p(q)}}, & \text{if } r(o \mid q) = 0. \end{cases}$$

GRPO-style objective We consider the expectation form of the GRPO-family objective (omitting KL regularization for clarity):

$$J_0(\theta) = \mathbb{E}_q \mathbb{E}_{o \sim \pi_{\text{old}}(\cdot|q)} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} f\left(\frac{\pi_\theta(o_t \mid q, o_{<t})}{\pi_{\text{old}}(o_t \mid q, o_{<t})},\ A(o \mid q) \right) \right].$$

Here, $f(x, y)$ is a surrogate objective function inherited from PPO-style policy optimization. For GRPO specifically,

$$f(x, y) = \min\big(xy, \text{clip}(x, 1 - \epsilon, 1 + \epsilon)\, y\big),$$

where x is a likelihood ratio and y is the advantage.

Structural assumption on f Following Li et al. (2025), we assume that $f(x, y)$ is non-decreasing in x and admits the decomposition

$$f(x, y) = \mathbf{1}\{y > 0\}\, y\, f^+(x, 1) \ - \ \mathbf{1}\{y \le 0\}\, |y|\, f^-(x, 1),$$

for some non-decreasing functions f^+ and f^-. This assumption holds for GRPO and its common variants, and separates the effect of the sign and magnitude of the advantage.

Decomposition by reward outcomes. Let $\pi^+_{\text{old}}(\cdot \mid q)$ and $\pi^-_{\text{old}}(\cdot \mid q)$ denote the conditional distributions of outputs given $r(o \mid q) = 1$ and $r(o \mid q) = 0$, respectively. Applying the law of total expectation and substituting the piecewise form of $A(o \mid q)$ yields

$$\begin{aligned} J_0(\theta) = \mathbb{E}_q \Bigg[& p(q) \sqrt{\frac{1 - p(q)}{p(q)}} \mathbb{E}_{o \sim \pi^+_{\text{old}}(\cdot|q)} \frac{1}{|o|} \sum_{t=1}^{|o|} f^+\left(\frac{\pi_\theta}{\pi_{\text{old}}}, 1 \right) \\ & - (1 - p(q)) \sqrt{\frac{p(q)}{1 - p(q)}} \mathbb{E}_{o \sim \pi^-_{\text{old}}(\cdot|q)} \frac{1}{|o|} \sum_{t=1}^{|o|} f^-\left(\frac{\pi_\theta}{\pi_{\text{old}}}, 1 \right) \Bigg]. \end{aligned}$$

Emergence of the variance-based weight Factoring out common terms, the objective simplifies to

$$J_0(\theta) = \mathbb{E}_q \sqrt{p(q)\big(1-p(q)\big)} \left[\mathbb{E}_{o\sim\pi^+_{\text{old}}(\cdot|q)} s_\theta(o,q) - \mathbb{E}_{o\sim\pi^-_{\text{old}}(\cdot|q)} s_\theta(o,q) \right],$$

where $s_\theta(o,q)$ denotes the corresponding token-averaged scoring function.

Therefore, each question q contributes to the objective with a multiplicative weight

$$\omega(q) = \sqrt{p(q)\big(1-p(q)\big)} = \mathrm{std}[r(o \mid q)].$$

Implication This derivation shows that GRPO-style objectives implicitly scale each question by the standard deviation of its reward distribution. As a result, questions that are nearly always correct ($p(q) \approx 1$) or nearly always incorrect ($p(q) \approx 0$) receive vanishing weight, while questions of intermediate difficulty are emphasized.