Can Large Reasoning Models Self-Train?

Sheikh Shafayat[1]* **Fahim Tajwar**[2]*
Ruslan Salakhutdinov[2] **Jeff Schneider**[2] **Andrea Zanette**[2]
[1] Independent Researcher
[2] Carnegie Mellon University

Abstract

Scaling the performance of large language models (LLMs) increasingly depends on methods that reduce reliance on human supervision. Reinforcement learning from automated verification offers an alternative, but it incurs scalability limitations due to dependency upon human-designed verifiers. Self-training, where the model's own judgment provides the supervisory signal, presents a compelling direction. We propose an online self-training reinforcement learning algorithm that leverages the model's self-consistency to infer correctness signals and train without any ground-truth supervision. We apply the algorithm to challenging mathematical reasoning tasks and show that it quickly reaches performance levels rivaling reinforcement-learning methods trained explicitly on gold-standard answers. Additionally, we analyze inherent limitations of the algorithm, highlighting how the self-generated proxy reward initially correlated with correctness can incentivize reward hacking, where confidently incorrect outputs are favored. Our results illustrate how self-supervised improvement can achieve significant performance gains without external labels, while also revealing its fundamental challenges.

1 Introduction

Pre-training large-scale neural networks on human-curated corpora (e.g., web-scale texts and carefully curated datasets) has endowed language models with broad general-purpose capabilities (Brown et al., 2020; Rae et al., 2022). However, the availability of human-generated data for pre-training is becoming a limiting factor, particularly as computational resources continue to scale rapidly (Hoffmann et al., 2022; Sevilla et al., 2022). Post-training methods, such as supervised finetuning, offer a way to rapidly improve model capabilities via a much smaller set of highly curated human-generated examples, but such reliance on high quality human-generated examples offers limited scalability (Ouyang et al., 2022; Bai et al., 2022).

In order to scale model capabilities further, reinforcement learning from human feedback can be used. This leverages human preference about model generated outputs, a type of feedback that is much easier to obtain (Ouyang et al., 2022; Kaufmann et al., 2024). Reinforcement learning with verifiable rewards offers a scalable alternative to human feedback by using automatic correctness checks as supervisory signals, an approach that has shown considerable success in reasoning and agentic tasks (DeepSeek-AI et al., 2025; OpenAI et al., 2024). However, this approach often depends on human-designed verifiers, which reintroduce the very scalability issues it seeks to overcome.

Achieving superhuman performance will ultimately require models that train in settings where even humans may not know the correct solutions, and are thus unable to provide feedback on model outputs. In such scenario, models must **self-improve**—that is, they must be able to assess the correctness of their own outputs and use their own judgment as a feedback signal to refine future

*Equal contribution.
Correspondence to: sheikhshafayat2@gmail.com, ftajwar@andrew.cmu.edu, zanette@cmu.edu

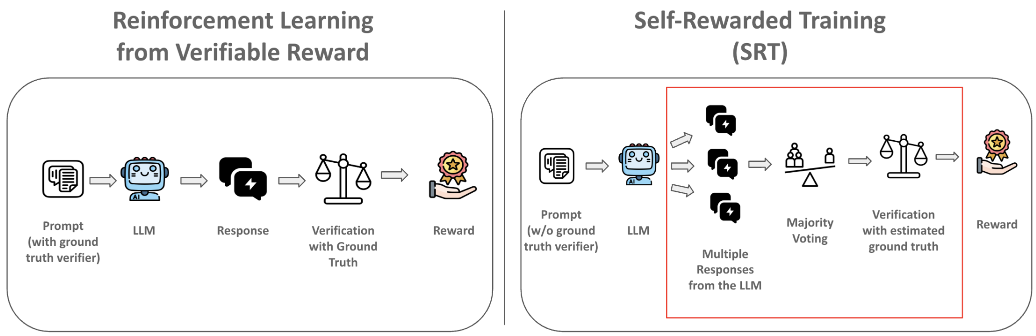

Figure 1: **(Overview of SRT)** In RLVR, one produces the reward for RL training using a ground truth verifier. Contrary to that, SRT does not assume access to a ground truth verifier; instead it uses majority voting from the model's own generations to estimate the ground truth, and use this proxy reward signal to train the model.

generations (Zelikman et al., 2022; Song et al., 2025; Huang et al., 2025). If this process can be sustained iteratively, models have the potential to enhance their performance continually. A particularly promising opportunity for such self-improvement arises in contexts characterized by a positive **generation-verification gap** (Song et al., 2025), where generating correct solutions is challenging, but verifying their correctness is relatively easy. This gap naturally arises in domains such as mathematics, games, and goal-based agentic tasks, where solutions often require complex reasoning or exploration, yet their correctness can be verified more easily.

In this work, we investigate self-improvement methods for large language models aimed at enhancing their mathematical reasoning abilities without reliance on ground-truth annotations or explicit rule-based verifiers. Specifically, we leverage an intrinsic verification criterion, namely *model self-consistency* (Wang et al., 2023a), to produce supervisory rewards, capitalizing on the observation that consistency among model-generated answers correlates positively with their correctness. Conceptually similar approaches, relying either directly or indirectly on model-generated self-consistency signals, have been explored previously (e.g., Huang et al. (2023); Prasad et al. (2024)) in mathematical domains. Typically, such approaches have been confined to one or few rounds of improvement. In contrast, the key contribution of this work is to demonstrate that this intrinsic supervisory signal can continuously drive model improvement within a reinforcement learning (RL) framework (Sutton et al., 1998), iteratively enhancing performance without access to external labels.

An appealing feature of our proposed approach is its reliance solely on unlabeled data. Because our method does not depend on external labels, it can also be naturally applied in a "test-time training" fashion. Such flexibility allows models to iteratively boost their performance at inference time. Concurrently to our work, test-time training has also recently been investigated by Zuo et al. (2025).

While the prospect of continual model self-improvement is appealing, it is important to recognize a significant limitation: the model's self-generated reward serves only as a **proxy** for underlying correctness. Such proxy rewards may give rise to the phenomenon of **reward hacking**, whereby the model learns to maximize its self-assigned reward by generating increasingly consistent but potentially incorrect answers. This unintended behavior can reduce or even reverse the initial beneficial correlation between confidence and correctness, ultimately constraining sustained improvement. Understanding the precise dynamics of this process is thus essential for further advancement.

In summary, our contributions are fourfold:

(1) Motivated by prior works based on consistency based self-improvement, we introduce a simple yet effective self-training *reinforcement learning* methodology, Self-Rewarded Training (SRT), that uses consistency across multiple model-generated solutions to estimate correctness during RL training, providing self-supervision signals without labeled data.

(2) We empirically demonstrate that at early training stages, SRT achieves performance rivaling standard reinforcement-learning methods trained explicitly on gold-standard answers.

(3) We analyze the limitations associated with self-generated rewards, revealing how the model's reward function, initially correlated with correctness, can degrade to reflect confidence rather than true accuracy, leading to the problem of reward hacking.

(4) We propose strategies to mitigate reward hacking, laying the groundwork for effective future approaches to sustaining continual model improvement.

Our code is publicly available here: `https://github.com/tajwarfahim/srt`. For other project-related assets such as constructed datasets, please see our project website at `https://self-rewarding-llm-training.github.io/`.

2 Preliminaries

Let π_θ denote a language model parameterized by parameters θ. Given a prompt x, the model produces a response $y = (y^1, y^2, \dots)$ 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 $r(y)$, whose dependence on x is omitted for brevity, extracts the model's proposed solution from the generated response and evaluates its correctness against the prompt-specific ground-truth answer:

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

Typically, one optimizes the *pass rate* defined as the average accuracy of the model π_θ across a distribution of prompts $\mathcal{X}$, namely

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

Taking the gradient of the objective function (3) with respect to θ and employing a baseline (for variance reduction) (Sutton et al., 1998) leads to the well-known policy gradient formulation:

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

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 pass rate for prompt x. In practice, the policy gradient in equation 4 is estimated through Monte Carlo samples, yielding the classical REINFORCE algorithm (Williams, 1992). Recent works have modified this base policy gradient formulation to improve its stability, efficiency, and practicality, resulting in advanced methods such as REINFORCE++ (Hu et al., 2025), GRPO (Shao et al., 2024), PPO (Schulman et al., 2017), and RLOO (Ahmadian et al., 2024a).

Implicit to our method is the notion of a **generation-verification gap** (Song et al., 2025), where generating correct solutions is hard, but verifying them is easy. Due to space constraints, we present the definition in Appendix A.

3 Self-Rewarded Training

In the absence of external supervision, language models must rely on themselves to produce supervisory signals. Intuitively, if the model can identify a higher-quality answer among several of its generations, this identified improvement can serve as a training signal. Such situations naturally arise in problems characterized by a positive generation-verification gap, including those in mathematics, logical reasoning, and code generation tasks (Li et al., 2022; Brown et al., 2025; Yu et al., 2024).

Self-supervision via majority voting. A simple yet effective method for leveraging this gap is *majority voting*, a technique empirically shown to achieve higher accuracy compared to individual model generations (Wang et al., 2023a). In our setting, majority voting involves: (1) sampling multiple answers per prompt, (2) grouping answers according to their parsed final solutions, and (3) estimating the ground truth answer with the most common solution (the mode). Prior works have demonstrated empirically that fine-tuning models directly on these majority-voted labels can boost performance (Huang et al., 2023; Prasad et al., 2024). However, these methods typically employ a single round of improvement guided by majority decisions, with performance bounded by the offline nature of the training procedure. Motivated by recent impressive performance improvements obtained through *reinforcement learning* in reasoning domains (OpenAI et al., 2024; DeepSeek-AI et al., 2025), we explore whether this self-supervision approach can be effectively extended and iteratively sustained within a reinforcement-learning loop.

Self-Rewarded Training (SRT). Motivated by these considerations, we propose framing self-improvement as a reinforcement learning problem, where labels are dynamically derived from the model's evolving majority votes. At a high level, each RL iteration proceeds as follows: **(1)** Sample a mini-batch of prompts and generate n answers per prompt using the current model, **(2)** Determine pseudo-labels by identifying the majority-voted answer for each prompt, **(3)** Use agreement with the majority-voted answer y_{majority} as an intrinsic binary reward:

$$r(y) = \mathbf{1}[\text{answer}(y) = y_{\text{majority}}], \tag{6}$$

(4) Perform a single RL update step on this mini-batch using the derived reward function $r(\cdot)$.

The complete approach is concisely detailed in Algorithm 1. Concretely, our method prescribes a specific form of the reward function using model self-consistency and it is thus compatible with all the common RL training algorithms such as PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024a), REINFORCE(Williams, 1992) and REINFORCE+++ (Hu et al., 2025). Furthermore, as the number of generations per prompt typically falls in the range 16-64 (Yu et al., 2025), SRT incurs no additional compute cost compared to the label-based version of these algorithms.

Algorithm 1: Self-Rewarded Training (SRT)

```
Input: Prompt dataset 𝒳
foreach RL iteration do
    /* Inference step */
    Sample minibatch ℬ ⊆ 𝒳
    foreach prompt x ∈ ℬ do
        Generate n solutions y^(1), ..., y^(n) ~ π_θ(·|x)
        Identify majority-vote answer:
            y_majority ← arg max_{y'} Σ_{i=1}^{n} 1[answer(y^(i)) = y']
        Define reward function:
            r(y) ← 1[answer(y) = y_majority]
    /* Gradient update step */
    Perform RL gradient update using r(·)
```

As long as majority voting leads to a positive generation-verification gap at each RL iteration, we expect iterative self-rewarding to provide a useful supervisory signal. We describe our observations about the empirical performance of SRT in the following section.

4 Experiments

We design experiments to systematically investigate the strengths and limitations of the SRT algorithm. More precisely, we aim to address the following questions: **(1)** How effective is SRT compared to standard reinforcement learning methods trained explicitly on ground-truth labels, and how well does this approach generalize to unseen problems? **(2)** Can self-improvement be sustained iteratively, leading to continuous performance enhancement, or is improvement ultimately bounded? **(3)** What underlying factors influence the effectiveness of self-improvement? **(4)** Since our approach does not require labeled data, it is applicable at both training and inference phases; thus, what is the effectiveness of SRT when used to increase performance at the test time? We describe our experimental setup first, and then report our observations.

Experimental setup. We use a Qwen2.5-Math-7B (Yang et al., 2024) model for all our experiments. Since our algorithm is independent of the choice of the underlying policy gradient algorithm, we choose RLOO (Ahmadian et al., 2024b; Kool et al., 2019) for its simplicity. We implement our algorithm on top of the `verl`[2] framework (Sheng et al., 2024). Unless otherwise mentioned, for every RL iteration, we sample 16 prompts, and for each prompt, we generate 32 rollouts from the policy to compute the majority voting label and subsequently use all such rollouts for training. Our KL loss coefficient is 0.001 unless mentioned otherwise. We use the AdamW (Loshchilov & Hutter, 2019) optimizer with 10^{-6} learning rate and 0.01 weight decay for all training runs, and usually do not use any learning rate decay. In practice, not all rollouts for a given prompt may contain a parsable solution. In such cases, we perform majority voting only among the rollouts from which a solution can be extracted. If no such rollouts exist, we discard the prompt and all its rollouts from the loss calculation.

Dataset. For most of the experiments, we use three training datasets: (1) DAPO (Yu et al., 2025) which contains 17398 math problems, primarily sourced from AoPS [3], a competitive math training platform, (2) MATH500 (Lightman et al., 2023), which contains 12,000 prompts for training and 500 prompts for validation from the original MATH dataset Hendrycks et al. (2021), (3) AIME 1983-2024 (Veeraboina, 2023), where we filter out the AIME-2024 prompts for validation and use the other 919 prompts for training.

4.1 Self-Training via Majority Voting

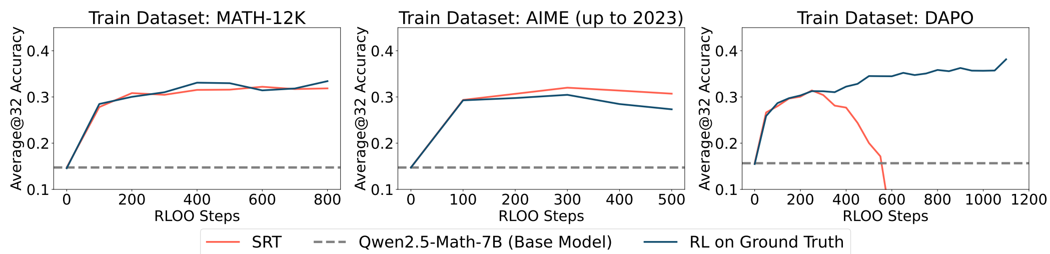

Figure 2: **(Self-Training Performance)** Average over 3 heldout test dataset performance of SRT vs training with ground truth reward. Training for longer with SRT can lead to performance collapse, but at the peak, it leads to around 100% performance improvement over the base model. Earlier in the training, it leads to similar improvement as regular RLOO training with ground truth.

We first investigate question **(1)**: How effective is self-improvement compared to standard RL methods trained explicitly on ground-truth labels, and how well does it generalize to heldout datasets? To address this, we train the base model on each of the training datasets (MATH, DAPO, AIME-1983-2023) and evaluate it on out-of-distribution settings using the following heldout test sets: AIME 24 (Jia, 2024), AIME 25 (Lin, 2025), and AMC (He, 2024). We aggregate performance metrics by averaging model accuracy (pass@1 rate, calculated using 32 rollouts per prompt) across these three distinct datasets to reduce variance. The results are presented in Figure 2; additional results, including the comparison with the baselines, can be found in Appendix C.

Surprisingly, we find that on the MATH and AIME training datasets, the self-supervised SRT method achieves results comparable to reinforcement learning on ground truth labels, despite having no access to ground-truth signals. Notably, the pass@1 scores reported in Figure 2 are evaluated on held-out datasets, demonstrating that our self-training procedure generalizes robustly beyond its training distribution.

The results observed for DAPO are more intricate, however. Specifically, when trained on DAPO, we observe that the performance of the SRT algorithm on the test set initially rises at a rate comparable to regular RL trained on ground-truth answers. Yet, approximately around 400-600 training steps, SRT reaches peak performance and subsequently starts to deteriorate. Regular RL training with ground truth labels, on the other hand, continues improving gradually beyond this point. We revisit and analyze this decline in performance in depth in the subsequent section (Section 4.1.1).

Overall, our findings reveal a striking and somewhat unexpected trend: **even without any labeled examples, the performance curve of SRT closely tracks that of RL on gold answers during the**

[2]`https://github.com/volcengine/verl`

[3]`https://artofproblemsolving.com/`

initial stages of training. Within statistical uncertainties, we observe that SRT essentially matches the peak test-time pass@1 scores of labeled RL methods when training on MATH and AIME'83-AIME'23 datasets. On the challenging DAPO dataset, SRT is still able to achieve a significant fraction (75%) of the final performance exhibited by RL. Moreover, SRT's peak performance exhibits a relative improvement of around 100% over the base model on all 3 training datasets.

Importantly, the **observed performance collapse can be addressed by early stopping** the training. Similar to supervised learning or standard RL training, a small validation set with ground-truth labels can be employed to reliably identify the optimal stopping point and retain the peak performance of SRT. We discuss avoiding model collapse in more depth in Section 5 and Appendix D.

4.1.1 What Happens After SRT Peaks in Performance?

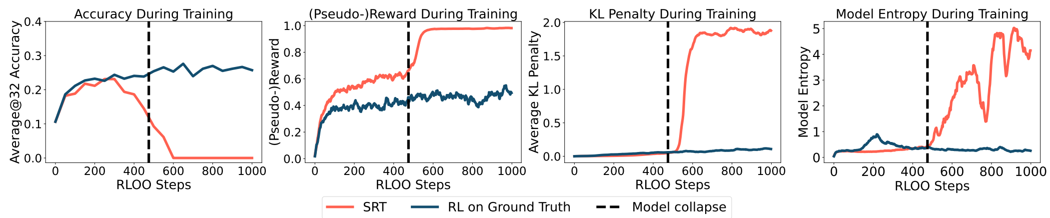

Figure 3: **(Self-Training Dynamics)** Extended training via SRT can lead to reward hacking, as demonstrated by the sudden hike in KL penalty and training (psuedo-)reward, but collapse of accuracy on the held-out test sets.

After SRT's performance peaks on the DAPO training dataset (see Figure 2), we observe that its test-time accuracy begins to deteriorate significantly. In fact, we find that a similar behavior also emerges when training on MATH-12k for more than two epochs, resulting in a noticeable performance collapse (see Appendix C for further details). To develop a clearer understanding of the underlying reasons for this phenomenon, in this section we investigate this model collapse closely.

First, we examine the training dynamics to ascertain whether the SRT self-training objective is being correctly optimized. We plot the running value of the SRT objective (Eqn. 6) in Figure 3. The observed performance collapse closely coincides with a sudden increase in the SRT self-reward objective, implying that the optimization procedure has, in fact, maximally optimized the training objective (self-consistency majority voting), despite a decline in *actual* output correctness. On the same figure, we further report the token-level average Kullback–Leibler divergence between the model under SRT training and the base model. We observe a sharp increase in KL divergence at the exact point when performance begins to deteriorate, indicating that the generative distribution of the model has substantially diverged from the original model.

These findings strongly suggest the occurrence of **reward hacking**—the model has learned to produce consistent responses in order to optimize its self-assigned reward, irrespective of their true correctness. Indeed, manual analysis of the model outputs (examples provided in Appnedix E) confirms this hypothesis: after collapse, the model outputs a very high entropy, essentially random, set of tokens followed by the same "template" final answer that is nearly independent of the input prompt. In other words, the initially strong correlation between the SRT objective and correctness is ultimately compromised, becoming no longer a reliable proxy signal.

Theoretical justification There is a simple yet precise theoretical justification for this behavior:

> *The reinforcement learning optimization problem defined by the* SRT *objective explicitly encourages consistency across outputs, independently of correctness.*

Consequently, an optimal policy under this objective is to produce an identical response irrespective of the input, thus artificially attaining maximal possible reward. Therefore, one should naturally expect that continued training under such a proxy objective can result in this degenerate solution, especially when optimizing for this is easier than learning to solve the actual task.

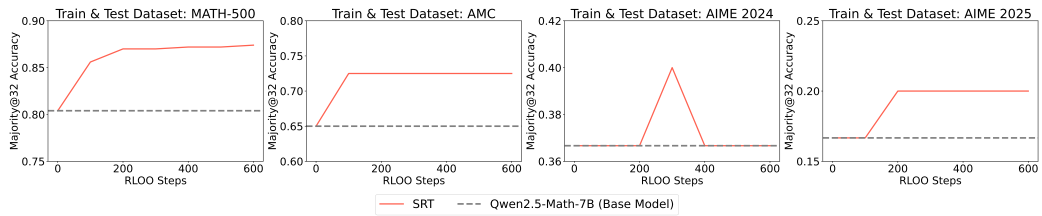

Figure 4: **(Test-Time Self-Training Performance)** Given the test dataset $\mathcal{D}_{\text{test}}$, one can perform SRT on $\mathcal{D}_{\text{test}}$ before making predictions. Our results show that this improves the majority voting performance on $\mathcal{D}_{\text{test}}$ without access to ground truth labels.

4.2 Test-Time Self-Improvement

An appealing application of self-training is improving model accuracy via test-time training (Sun et al., 2020; Wang et al., 2021), a direction also explored by the concurrent work of Zuo et al. (2025). Test-time training refers to the procedure of further adapting or fine-tuning a pre-trained model on the actual test set itself, typically without access to labels or ground truth annotations. Applying SRT as a test-time training technique is remarkably straightforward: the unlabeled test set is treated precisely as if it were a training dataset, and SRT is directly applied.

We compare the test-time performance of majority voting after SRT test-time training as well as without any test-time training. Empirically, we observe (Fig 4) that test-time training via SRT provides relatively limited, yet noticable, performance gains when measured under the maj@32 metric, compared to the popular majority voting baseline applied directly to outputs generated by the base model. Moreover, the performance gain over majority voting from the base model is larger on larger test datasets.

4.2.1 Why Doesn't the Performance Collapse during Test-Time-Training?

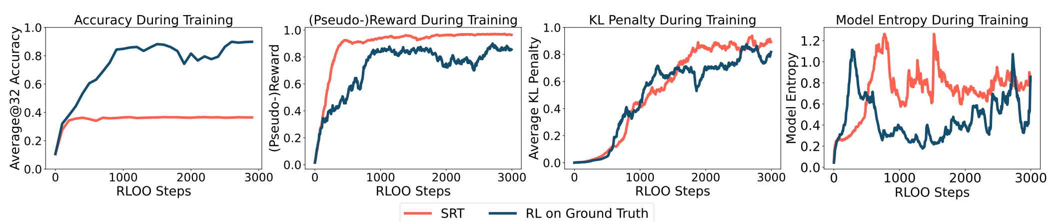

Figure 5: **(Test-Time Self-Training Dynamics)** We apply test-time training on AIME 2024 and observe no performance collapse. However, SRT's performance quickly saturates (leftmost plot), and the pseudo-reward value (second plot) also approaches saturation.

Interestingly, upon completion of test-time training, a visual inspection of model's outputs reveals that the model's predictions still degenerate to a single response for nearly every test prompt—precisely the behavior identified as optimal solution to the SRT objective; however, the test-time accuracy remains high.

We conjecture that test-time self-training is inherently more stable due to crucial differences in dataset size. For example, consider the AIME24 test dataset, which contains only 30 samples for self-improvement. With such a limited sample size, the model quickly converges to a stable majority vote answer on these examples by reinforcing the particular chain-of-thought reasoning that leads to such solutions. After reaching this convergence, SRT ceases to receive meaningful gradient signals for further parameter updates, naturally stabilizing test-time performance (see Figure 5 for test-time training dynamics).

In contrast, during regular training on large-scale datasets, the iterative supply of many fresh samples continually pushes the model to optimize heavily for consistency. In such conditions, the model is incentivized to adopt an overly simplistic generalization strategy (producing same `\boxed{}` answer)—eventually collapsing by producing a uniform, prompt-independent prediction.

5 Can Model Collapse be Avoided?

As discussed in Section 4.1.1, the optimization objective in SRT can lead to significant initial improvements followed by eventual model collapse. Here, we explore complementary strategies to address model collapse and further enhance the performance achievable via self-training:

(1) An *early stopping* strategy leveraging a small labeled validation dataset to detect and prevent model collapse.

(2) An *algorithmic* strategy that mitigates the risk of model collapse by using pseudo-labels generated from a stable base model rather than from the continuously updated model.

(3) A *data-driven* curriculum-based strategy to enhance model performance beyond simple early stopping.

Additional results are discussed in Appendix D.

5.1 Early Stopping

Using a validation set for hyperparameter tuning and model selection is a well-established best practice. In our context, even a small labeled validation dataset can effectively identify the peak performance point during self-training, thereby mitigating model collapse. Figure 6 shows the progression of model performance, measured throughout training on the DAPO dataset and evaluated across several test sets. Crucially, we find that the peak performance consistently occurs around the same training step across different held-out datasets. This observation suggests that any of these datasets can serve effectively for early stopping. Concretely, the vertical line in Figure 6 illustrates early stopping based on just 1% of the DAPO dataset held out as a validation set. The model performance on all other evaluation datasets at this point remains close to optimal.

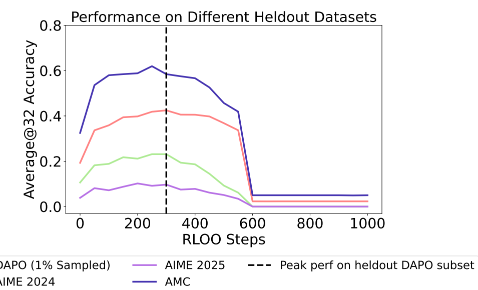

Figure 6: **(Early Stopping is Effective)** The peak performance occur at nearly the same point for all heldout sets, so using any would be effective for early stopping.

5.2 Self-Training with Offline-Generated Labels

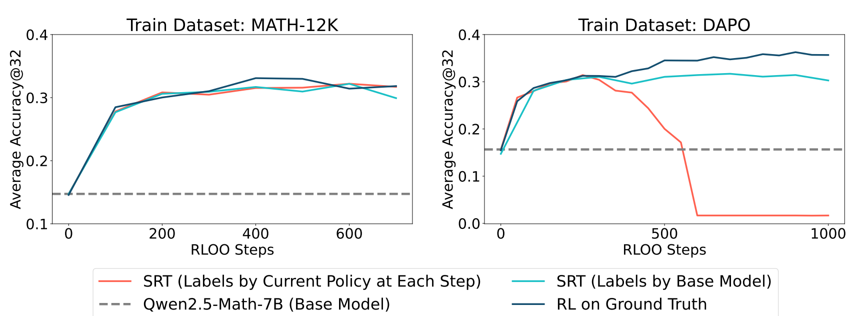

Figure 7: **(Self-Training with Offline-Generated Labels)** Performance comparison between SRT and a baseline variant, where pseudo-labels are precomputed from a fixed base model checkpoint. The offline-generated labels maintain training stability while achieving comparable performance to SRT, highlighting a limitation of the online labeling strategy.

As identified in Section 4.1.1, the tendency toward model collapse arises because SRT prioritizes consistency over correctness, increasingly rewarding model agreement even if incorrect. A straightforward yet effective approach to address this involves generating pseudo-labels from a stable, previously fixed checkpoint, rather than leveraging labels from the evolving policy. To evaluate this approach, we generate pseudo-labels via majority voting rollouts from the Qwen2.5-Math-7B base model, store these offline-generated labels, and subsequently perform RL training against them. Figure 7 demonstrates that training with these offline-generated labels significantly stabilizes training while achieving performance comparable to SRT. Importantly, this indicates that the dynamic updating of pseudo-labels during training (online labeling) may not always confer substantial benefits, and instead can contribute to training instability.

5.3 Self-Training with Curriculum Learning

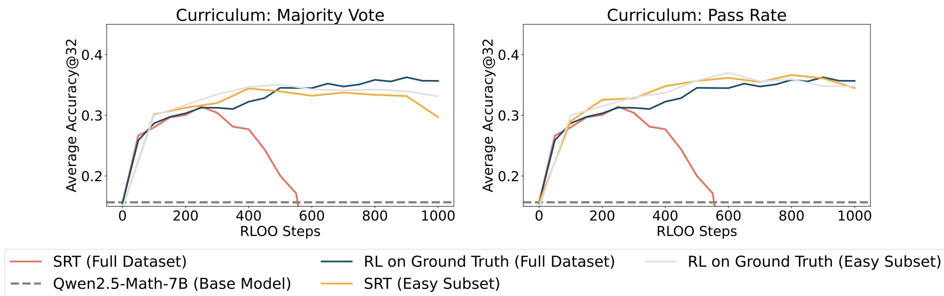

Figure 8: **(Curriculum-Based Self-Training)** Performance of SRT on curated subsets containing the easiest 1/3 of prompts from the DAPO dataset, selected based either on model pass rate or frequency of the majority vote. Training on these easier subsets prevents reward hacking even after extensive training (3 epochs), demonstrating the effectiveness of curriculum learning strategies in sustaining continual model improvement.

We hypothesize that *model collapse occurs more rapidly when training on more challenging datasets*. This conjecture aligns with our empirical findings in Section 4.1.1: the model experiences earlier collapse when training on the difficult DAPO dataset compared to the simpler MATH-12K dataset. The intuition is that, on a more challenging dataset, the model finds it easier to abandon its pre-trained knowledge in favor of optimizing self-consistency rather than genuinely learning to solve the underlying task.

We leverage this hypothesis to implement a curriculum learning strategy (Bengio et al., 2009; Andrychowicz et al., 2017; Portelas et al., 2020; Florensa et al., 2017; Song et al., 2025; Lee et al., 2025; Tajwar et al., 2025) by identifying the 'easiest' subset of the DAPO dataset. To be precise, we retain 1/3-rd of the easiest DAPO prompts selected according to two distinct metrics:

(1) **Pass rate of the base model**, which utilizes ground-truth labels.

(2) **Frequency of the majority vote**, which does not require ground-truth labels.

Figure 8 shows that training on these easier subsets significantly delays the onset of reward hacking, allowing for continuous improvement even across multiple epochs. Remarkably, performance on these curriculum subsets reaches levels comparable to standard RL training with ground-truth labels on the entire DAPO dataset. These promising results suggest that curriculum strategies may further extend the benefits of SRT, opening exciting avenues for future investigation.

6 Related Works

Self Improving LLM. Previous works (Zelikman et al., 2022; Wang et al., 2023b; Huang et al., 2023; Madaan et al., 2023; Chen et al., 2024; Gulcehre et al., 2023; Singh et al., 2024; Ni et al., 2023; Hwang et al., 2024; Havrilla et al., 2024; Pang et al., 2024) have demonstrated the feasibility of LLMs' self-improvement over their previous iteration by training on data distilled by the previous instances of the model. Most of these approaches usually have data filtering/reranking step in the pipeline, which is often performed by the model itself (Wu et al., 2025) or by training another (Hosseini et al., 2024) verifier model. Particularly, (Huang et al., 2023; Wang et al., 2023a; Prasad et al., 2024) demonstrated the feasibility of using majority voting and self-consistency to filter chain-of-thought traces that, when used as SFT training data, improve the LLM performance on downsteaming tasks. A concurrent work, (Zhao et al., 2025), proposes a self-evolution, self-play pipeline where an LLM generates coding problems of appropriate difficulty, solves and trains on them using RLVR. The model-generated solutions are validated by a code executor in the loop. To the best of our knowledge, our work is the first to explore *online RLVR's* potential in self improving LLMs for math specific reasoning as opposed to a single round of SFT or DPO (Prasad et al., 2024; Huang et al., 2023). In a recent work, (Song et al., 2025) formalized a generation verification gap as central to the model's ability to self-improve. Similarly, (Huang et al., 2025) proposed a "sharpening" mechanism as the key to self-improvement. Our SRT pipeline builds on top of both of these intuitions.

Online RLVR and Easy to Hard Generalization. Online reinforcement learning with verifiable reward (RLVR) (Lambert et al., 2025) has emerged as a new paradigm of LLM post-training especially for enhancing math, coding and reasoning performances (OpenAI et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025; Lambert et al., 2025). Despite the success of the reasoning models, it is still unclear to what extent they can generalize beyond the difficulty of their training data distribution, a problem termed easy to hard generalization (Sun et al., 2024). (Sun et al., 2024) shows that models can be trained to solve level 4-5 MATH(Hendrycks et al., 2021) problems after training using a process reward model trained on MATH level 1-3 dataset. Another work (Lee et al., 2025) explores this question and finds that transformers are capable of easy-to-hard generalization by utilizing *transcendence* phenomenon (Zhang et al., 2024) in the context of simple addition, string copying, and maze solving using small language models.

Model Collapse and Reward Hacking. Model collapse is a well-known phenomenon in training on self-generated training data (Alemohammad et al., 2024; Shumailov et al., 2024b,a; Bertrand et al., 2024; Briesch et al., 2025), and multiple approaches related to data mixing, reliable verification, training using contrastive loss using negative samples and curriculum learning have been proposed (Gerstgrasser et al., 2024; Feng et al., 2025; Briesch et al., 2025; Song et al., 2025; Gillman et al., 2024; Setlur et al., 2024) to prevent models from collapsing, which previous work on LLM's easy to hard generalization (Lee et al., 2025) also utilize. However, in RL paradigm, we do not directly do supervised fine-tuning on model-generated data, and it remains an open question to what extent the previous findings of model collapse apply to our RLVR setting. In our work, we show that models trained using RL on self-labeled data often suffer from actor collapsing due to reward hacking (Amodei et al., 2016; Denison et al., 2024) and propose a few strategies to mitigate it.

Data Efficient RLVR. Several concurrent works look into the data efficiency of the RLVR pipeline. Notably, (Wang et al., 2025) shows that by just training on one example, the model can achieve performance equivalent to training on 1.2k examples from the DeepScaleR dataset (Luo et al., 2025). Similarly, TTRL (Zuo et al., 2025), also proposes a label-free online RLVR paradigm in a test-time setting, similar to SRT. Our work also explores this paradigm in Section 4.2, however, we focus on reducing the need for ground truth labels instead of training examples.

7 Conclusion

In this work, we introduced Self-Rewarded Training (SRT), a reinforcement learning approach enabling continual self-improvement of language models through intrinsic self-consistency, eliminating the need for external annotations. Our experiments demonstrated promising initial performance comparable to models trained with explicit ground-truth signals. However, we also identified significant limitations, notably reward hacking, where models optimize consistency rather than correctness, risking eventual performance collapse. We proposed preliminary mitigation strategies such as early stopping and curriculum learning to address this issue.

Further investigations could explore the role of the base model's as well as develop more robust forms of self-verification. Additionally, integrating labeled data in a semi-supervised RL framework or exploring curriculum learning techniques are promising directions. We leave these as well as other exciting directions towards self-improving models for future work.

Acknowledgements

This work has greatly benefited from the use of Delta's advanced computing and data resource supported by the National Science Foundation (OAC 2005572) and the State of Illinois, as part of ACCESS-approved compute grants (Boerner et al., 2023). We also appreciate the computing resources of Bridges-2 (Brown et al., 2021) at Pittsburgh Supercomputing Center through ACCESS allocation CIS240901 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. Overall, this project used ACCESS grants CIS240901, CIS230366, CIS250216, and CIS250428 for its compute resources. Moreover, FT was partially supported by the U.S. Army Futures Command under Contract No. W519TC-23-C-0030 during the project.

The authors thank Brandon Pusateri, Jillian Lehosky, and Greg Bauer from ACCESS Support Staff for their incredible help at approving supplements and renewals for ACCESS compute grants throughout this project. Moreover, the work would not have finished so quickly without the help of Brett Bode from NCSA Delta Support Staff, who provided the authors with critical help in properly utilizing the Delta cluster. FT gratefully acknowledges Daman Arora, Yuda Song, Yiding Jiang, Ruiqi Zhang, Zhaoyi Zhou, Guanning Zeng, Yutong He and other members of the Zanette, Russ, and Auton lab for feedback and suggestions received on earlier versions of this work. FT also greatly benefited from using Prof. Aviral Kumar's lab's computational resources for initial implementation and debugging of the algorithm.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL `https://aclanthology.org/2024.acl-long.662/`.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024b. URL `https://arxiv.org/abs/2402.14740`.

Alemohammad, S., Casco-Rodriguez, J., Luzi, L., Humayun, A. I., Babaei, H., LeJeune, D., Siahkoohi, A., and Baraniuk, R. Self-consuming generative models go MAD. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=ShjMHfmPs0`.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety, 2016. URL `https://arxiv.org/abs/1606.06565`.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL `https://arxiv.org/abs/2204.05862`.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Bertrand, Q., Bose, J., Duplessis, A., Jiralerspong, M., and Gidel, G. On the stability of iterative retraining of generative models on their own data. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=JORAfH2xFd`.

Boerner, T. J., Deems, S., Furlani, T. R., Knuth, S. L., and Towns, J. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, PEARC '23, pp. 173–176, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399852. doi: 10.1145/3569951.3597559. URL `https://doi.org/10.1145/3569951.3597559`.

Briesch, M., Sobania, D., and Rothlauf, F. Large language models suffer from their own output: An analysis of the self-consuming training loop, 2025. URL https://openreview.net/forum?id=SaOxhcDCM3.

Brown, B., Juravsky, J., Ehrlich, R. S., Clark, R., Le, Q. V., Re, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling, 2025. URL https://openreview.net/forum?id=0xUEBQV54B.

Brown, S. T., Buitrago, P., Hanna, E., Sanielevici, S., Scibek, R., and Nystrom, N. A. Bridges-2: A platform for rapidly-evolving and data intensive research. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions*, PEARC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382922. doi: 10.1145/3437359.3465593. URL https://doi.org/10.1145/3437359.3465593.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning convertsweak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Denison, C., MacDiarmid, M., Barez, F., Duvenaud, D., Kravec, S., Marks, S., Schiefer, N., Soklaski, R., Tamkin, A., Kaplan, J., Shlegeris, B., Bowman, S. R., Perez, E., and Hubinger, E. Sycophancy to subterfuge: Investigating reward-tampering in large language models, 2024. URL https://arxiv.org/abs/2406.10162.

Feng, Y., Dohmatob, E., Yang, P., Charton, F., and Kempe, J. Beyond model collapse: Scaling up with synthesized data requires verification. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MQXrTMonT1.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Gerstgrasser, M., Schaeffer, R., Dey, A., Rafailov, R., Korbak, T., Sleight, H., Agrawal, R., Hughes, J., Pai, D. B., Gromov, A., Roberts, D., Yang, D., Donoho, D. L., and Koyejo, S. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5B2K4LRgmz.

Gillman, N., Freeman, M., Aggarwal, D., Hsu, C.-H., Luo, C., Tian, Y., and Sun, C. Self-correcting self-consuming loops for generative model training, 2024. URL https://arxiv.org/abs/2402.07087.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., Macherey, W., Doucet, A., Firat, O., and de Freitas, N. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Havrilla, A., Raparthy, S., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., and Raileanu, R. Glore: When, where, and how to improve llm reasoning via global and local refinements, 2024. URL https://arxiv.org/abs/2402.10963.

He, Z. Amc23 dataset, 2024. URL https://huggingface.co/datasets/zwhe99/amc23.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022. URL `https://arxiv.org/abs/2203.15556`.

Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=stmqBSW2dV`.

Hu, J., Liu, J. K., and Shen, W. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025. URL `https://arxiv.org/abs/2501.03262`.

Huang, A., Block, A., Foster, D. J., Rohatgi, D., Zhang, C., Simchowitz, M., Ash, J. T., and Krishnamurthy, A. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=WJaUkwci9o`.

Huang, J., Gu, S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL `https://aclanthology.org/2023.emnlp-main.67/`.

Hwang, H., Kim, D., Kim, S., Ye, S., and Seo, M. Self-explore: Enhancing mathematical reasoning in language models with fine-grained rewards. In *EMNLP (Findings)*, pp. 1444–1466, 2024. URL `https://aclanthology.org/2024.findings-emnlp.78`.

Jia, M. Aime 2024 dataset, 2024. URL `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`.

Kaufmann, T., Weng, P., Bengs, V., and Hüllermeier, E. A survey of reinforcement learning from human feedback, 2024. URL `https://arxiv.org/abs/2312.14925`.

Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! In *DeepRL-StructPred@ICLR*, 2019. URL `https://api.semanticscholar.org/CorpusID:198489118`.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., Gu, Y., Malik, S., Graf, V., Hwang, J. D., Yang, J., Bras, R. L., Tafjord, O., Wilhelm, C., Soldaini, L., Smith, N. A., Wang, Y., Dasigi, P., and Hajishirzi, H. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL `https://arxiv.org/abs/2411.15124`.

Lee, N., Cai, Z., Schwarzschild, A., Lee, K., and Papailiopoulos, D. Self-improving transformers overcome easy-to-hard and length generalization challenges, 2025. URL `https://arxiv.org/abs/2502.01612`.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alphacode. *Science*, 378 (6624):1092–1097, 2022.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Lin, Y.-T. Aime 2025 dataset, 2025. URL `https://huggingface.co/datasets/yentinglin/aime_2025`.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W., Roongta, M., Cai, C., Luo, J., Zhang, T., Li, E., Popa, R. A., and Stoica, I. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Ni, A., Inala, J. P., Wang, C., Polozov, A., Meek, C., Radev, D., and Gao, J. Learning math reasoning from self-sampled correct and partially-correct solutions. In *ICLR*, 2023. URL `https://openreview.net/forum?id=4D4TSJE6-K`.

OpenAI, :, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., et al. Openai o1 system card, 2024. URL `https://arxiv.org/abs/2412.16720`.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL `https://arxiv.org/abs/2203.02155`.

Pang, J.-C., Wang, P., Li, K., Chen, X.-H., Xu, J., Zhang, Z., and Yu, Y. Language model self-improvement by reinforcement learning contemplation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=38E4yUbrgr`.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Prasad, A., Yuan, W., Pang, R. Y., Xu, J., Fazel-Zarandi, M., Bansal, M., Sukhbaatar, S., Weston, J., and Yu, J. Self-consistency preference optimization, 2024. URL `https://arxiv.org/abs/2411.04109`.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., et al. Scaling language models: Methods, analysis & insights from training gopher, 2022. URL `https://arxiv.org/abs/2112.11446`.

Schulman, J. Approximating kl divergence. `http://joschu.net/blog/kl-approx.html`, Mar 2020. Accessed: 2025-05-20.

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

Setlur, A., Garg, S., Geng, X., Garg, N., Smith, V., and Kumar, A. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *Advances in Neural Information Processing Systems*, 37:43000–43031, 2024.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. Compute trends across three eras of machine learning, 2022.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., and Anderson, R. The curse of recursion: Training on generated data makes models forget, 2024a. URL `https://arxiv.org/abs/2305.17493`.

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R. J., and Gal, Y. Ai models collapse when trained on recursively generated data. *Nat.*, 631(8022):755–759, July 2024b. URL `https://doi.org/10.1038/s41586-024-07566-y`.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A. T., Kumar, A., Alemi, A. A., Rizkowsky, A., Nova, A., Adlam, B., Bohnet, B., Elsayed, G. F., Sedghi, H., Mordatch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J., Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp, L. A., Xiao, L., Bileschi, M., Constant, N., Novak, R., Liu, R., Warkentin, T., Bansal, Y., Dyer, E., Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=lNAyUngGFK. Expert Certification.

Song, Y., Zhang, H., Eisenach, C., Kakade, S. M., Foster, D., and Ghai, U. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtJSMcF3ek.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. Test-time training for out-of-distribution generalization, 2020. URL https://openreview.net/forum?id=HyezmlBKwr.

Sun, Z., Yu, L., Shen, Y., Liu, W., Yang, Y., Welleck, S., and Gan, C. Easy-to-hard generalization: Scalable alignment beyond human supervision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qwgfh2fTtN.

Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Tajwar, F., Jiang, Y., Thankaraj, A., Rahman, S. S., Kolter, J. Z., Schneider, J., and Salakhutdinov, R. Training a generally curious agent, 2025. URL https://arxiv.org/abs/2502.17543.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., Tang, C., Wang, C., Zhang, D., Yuan, E., Lu, E., Tang, F., Sung, F., Wei, G., Lai, G., Guo, H., et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Veeraboina, H. Aime problem set 1983-2024, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., Chen, W., Wang, S., Du, S. S., and Shen, Y. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Wu, T., Yuan, W., Golovneva, O., Xu, J., Tian, Y., Jiao, J., Weston, J. E., and Sukhbaatar, S. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge, 2025. URL `https://openreview.net/forum?id=lbj0i29Z92`.

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

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., et al. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Yu, L., Jiang, W., Shi, H., YU, J., Liu, Z., Zhang, Y., Kwok, J., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=N8N0hgNDRt`.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Dai, W., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Qiao, M., Wu, Y., and Wang, M. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL `https://arxiv.org/abs/2503.14476`.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zhang, E., Zhu, V., Saphra, N., Kleiman, A., Edelman, B. L., Tambe, M., Kakade, S. M., and eran malach. Transcendence: Generative models can outperform the experts that train them. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=eJG9uDqCY9`.

Zhao, A., Wu, Y., Yue, Y., Wu, T., Xu, Q., Yue, Y., Lin, M., Wang, S., Wu, Q., Zheng, Z., and Huang, G. Absolute zero: Reinforced self-play reasoning with zero data, 2025. URL `https://arxiv.org/abs/2505.03335`.

Zuo, Y., Zhang, K., Qu, S., Sheng, L., Zhu, X., Qi, B., Sun, Y., Cui, G., Ding, N., and Zhou, B. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Definition of Generation-Verification Gap

The single-generation accuracy is defined as:

$$\text{Acc}_{\text{gen}}(\theta) = \mathbb{E}_{x\sim\mathcal{X}, y\sim\pi_\theta(\cdot|x)}[\mathbf{1}(y = y^*)],$$

where y^* is the correct solution. A verifier function f selects one candidate from multiple generations:

$$f\left(x, \{y^{(1)}, \dots, y^{(n)}\}\right) \in \{y^{(1)}, \dots, y^{(n)}\}.$$

We define verification accuracy as:

$$\text{Acc}_{\text{ver}}(\theta, n) = \mathbb{E}_{x\sim\mathcal{X}}\left[\mathbf{1}\left(f\left(x, \{y^{(1)}, \dots, y^{(n)}\}\right) = y^*\right)\right].$$

We say that a positive *generation-verification gap* occurs whenever $\text{Acc}_{\text{ver}}(\theta, n) > \text{Acc}_{\text{gen}}(\theta)$. Such a gap indicates the verifier's greater proficiency in recognizing correct solutions within a set of candidates compared to the generator independently generating correct answers.

A.1 Generation Verification Gap Through Majority Voting

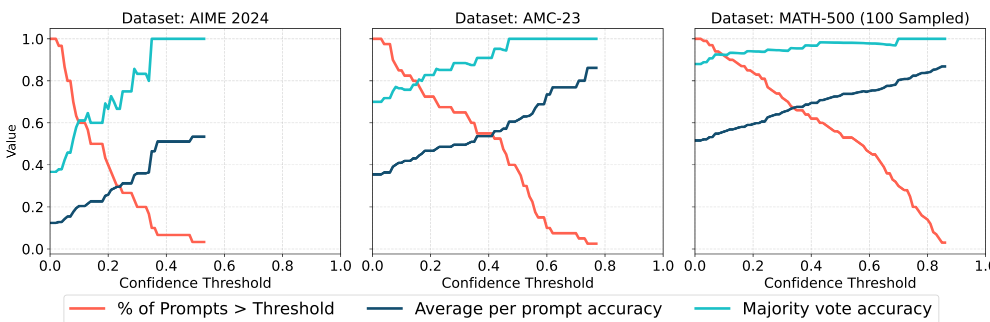

Figure 9: Three of our test datasets display evidence of generation verification gap through majority voting. **The positive gap between majority voting accuracy and per prompt accuracy means LLMs can utilize this gap as a learning signal to improve their average accuracy**. x axis refers to the threshold cut off self-consistency (proportion of answers that are majority voted answers). Higher x value refers to more self-consistent LLM outputs. For example, at `x = 0.4`, we only keep LLM responses where at least 40% of the (properly parsed) responses were the majority voted answer. On y axis, **average per prompt accuracy** refers to the average number of correct answers out of the (successfully parsed) responses, across 32 generations per question (computed on a per prompt basis and then averaged over the dataset). **Majority vote accuracy** refers to how often the majority vote is the correct answer for that prompt (then averaged over the dataset). **% of Prompts** was computed over all 32 generations (*not* on the parsable answers for a fair comparison). If there is no prompt left over a certain threshold, the line plot ends there.

B Details on Implementation and Training

B.1 RLOO Algorithm

All experiments in this work use RLOO(Ahmadian et al., 2024a; Kool et al., 2019) for its simplicity.

$$\frac{1}{k}\sum_{i=1}^{k}[R(y_{(i)},x)-\frac{1}{k-1}\sum_{j\neq k}R(y_{(j)},x)]\nabla\log\pi(y_{(i)}|x) \text{ for } y_{(1)},...,y_{(k)} \overset{i.i.d}{\sim} \pi_\theta(\cdot|x)$$

Here k is the number of online samples generated. RLOO creates a dynamic baseline for each sample without needing a separate value function, effectively estimating the expected return on-the-fly during training. Not having a value networks makes the training much simpler and (Ahmadian et al., 2024a) shows it to be a very strong baseline. For all experiments in this paper, we set k=32.

In our implementation, we did not add KL penalty to the loss function, rather to the reward itself. In verl framework, this can be configured using `algorithm.use_kl_in_reward=True` and `actor_rollout_ref.actor.use_kl_loss=False`. To estimate KL penalty, we use the low variance KL estimator proposed by Schulman (2020):

$$\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) \approx \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} - 1 - \log\left(\frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right)$$

Sampling For all experiments, we kept the generation `temperature` to `1.0`, `top_k` and `top_p` to `-1` for both RL rollouts and evaluation. We cut off maximum prompt length at 1024 and maximum response length to 3072, since Qwen2.5-Math-7B models support a maximum context window of 4096.

B.2 GPU Infrastructure

All experiments in this work were conducted using either a single node consisting of 8 NVIDIA H200 GPUs (141 GB of GPU memory per GPU) or a single node consisting of 4 NVIDIA GH200 GPUs (96 GB of GPU memory per GPU). All experiments can be replicated in single-node training, and we did not, in fact, utilize multinode training. In total, this work consumed $\sim$8000 GPU hours (including preliminary studies and failed runs). All the final results listed in this paper can be replicated within 1000 H200 GPU hours.

B.3 Code and Dataset Release

For the ease of reproducing the results in this paper, we release the code and datasets used in this work. Specifically, we release four datasets (the others being publicly available):

1. Deduplicated version of the DAPO dataset, containing only unique prompts: `https://huggingface.co/datasets/ftajwar/deduplicated_dapo_dataset`
2. The test datasets compiled together: `https://huggingface.co/datasets/ftajwar/srt_test_dataset`
3. The easy subset of DAPO, chosen based on per prompt pass rate calculated using 32 responses for each prompt from a Qwen2.5-Math-7B model (at temperature 1.0 and no top_p or top_k sampling): `ftajwar/dapo_easy_one_third_sorted_by_pass_rate`
4. The easy subset of DAPO, chosen based on per prompt frequency of the majority answer from a Qwen2.5-Math-7B model using the same generation setting as above: `https://huggingface.co/datasets/ftajwar/dapo_easy_one_third_sorted_by_pass_rate`

The easy DAPO subsets are the most important datasets, since they allow quick reproduction of our results in Section 5.3. Finally, our code is available at: `https://github.com/tajwarfahim/srt`. For a summary of project related assets, visit our project website at `https://self-rewarding-llm-training.github.io/`.

C Additional Experimental Results

C.1 Performance on Individual Test Datasets

In Figure 2, we showed the performance of SRT vs training with ground truth reward, averaged across 3 heldout test sets. Now we present their performance on each individual test set for the sake of completeness. Figures 10, 11, and 12 show the detailed results when we train on DAPO, MATH-12K, and AIME (1983-2023) respectively. Additionally, when training on MATH-12K, we also evaluate the intermediate checkpoints on the heldout set MATH-500, which is reported in Figure 11. Since MATH-500 contains 500 examples, calculating average@32 accuracy becomes expensive, and hence we could not use it as a test set for the other training setups.

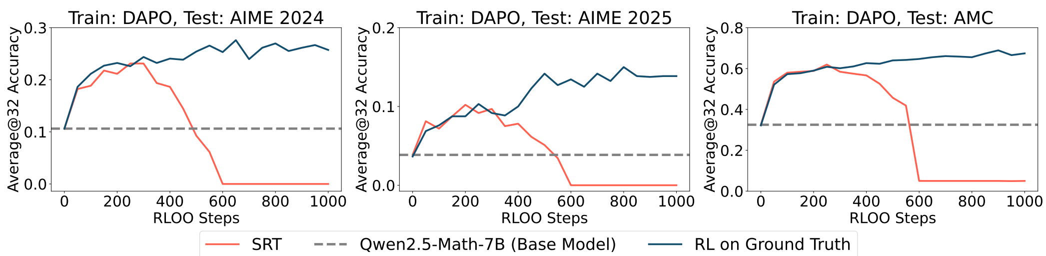

Figure 10: **(Individual test set performance during training on DAPO)** We record the average@32 accuracy during training Qwen2.5-Math-7B on DAPO, on three heldout test sets: AIME 2024, AIME 2025 and AMC. In all three cases, SRT performance collapses, while training with ground truth keeps improving steadily.

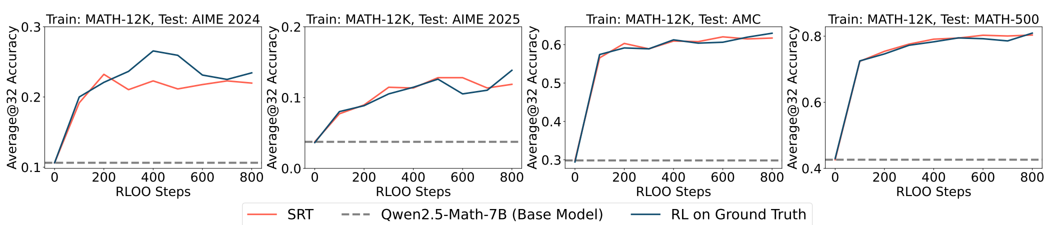

Figure 11: **(Individual test set performance during training on MATH-12K)** We record the average@32 accuracy during training Qwen2.5-Math-7B on MATH-12K, on three heldout test sets: AIME 2024, AIME 2025 and AMC. We also evaluate intermediate checkpoints on MATH-500 since we are training on MATH-12K (we could not do this for other training datasets due to a lack of computational resources). In all 4 heldout test sets, SRT results in similar performance gain as one would obtain from training with ground truth labels.

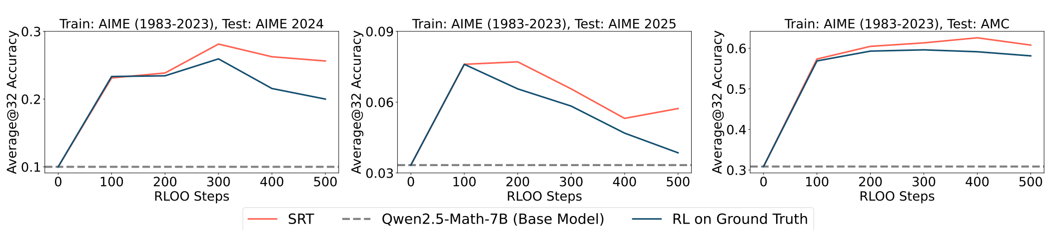

Figure 12: **(Individual test set performance during training on AIME (1983-2023))** We record the average@32 accuracy during training Qwen2.5-Math-7B on AIME (1983-2023), on three heldout test sets: AIME 2024, AIME 2025 and AMC. SRT performs similarly or better compared to training with ground truth labels over 10 epochs of training.

C.2 Experimental Results Using a Different Base Model

In addition to Qwen2.5-Math-7B, we apply our algorithm with another LLM — namely Qwen3-14B-Base (Yang et al., 2025). We choose the base model since it has not gone through additional post-training on reasoning tasks, unlike the Qwen3-14B model. Additionally, this is a significantly larger model with a different pre-training, making it suitable for testing our algorithm's effectiveness.

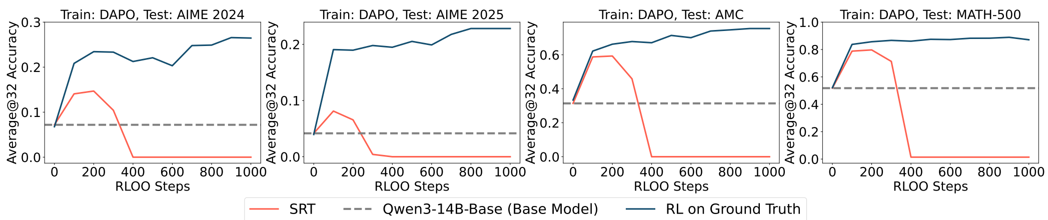

Figure 13: **(Individual test set performance during Qwen3-14B-Base on DAPO)** We record the average@32 accuracy during training a Qwen3-14B-Base model on DAPO, on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

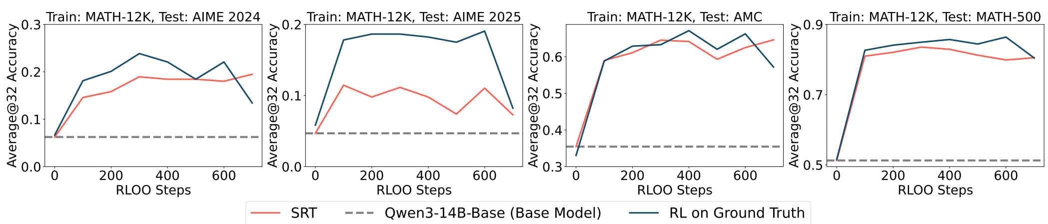

Figure 14: **(Individual test set performance during training Qwen3-14B-Base on MATH-12K)** We record the average@32 accuracy during training a Qwen3-14B-Base model on MATH-12K, on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

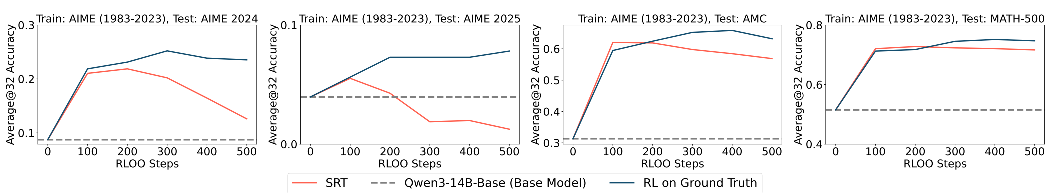

Figure 15: **(Individual test set performance during training Qwen3-14B-Base Model on AIME (1983-2023)** We record the average@32 accuracy during training a Qwen3-14B-Base model on AIME (1983-2023), on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

Figures 13, 14, and 15 shows our results with DAPO, MATH-12K, and AIME (1983-2023) used as training dataset respectively. Our experiments with Qwen3-14B-Base mostly follows similar patterns as Qwen2.5-Math-7B: SRT maintains stable performance on MATH-12K, mixed results on AIME (1983-2023), and performance collapse on DAPO.

D More on Enhancing SRT's Performance

D.1 Training Dynamics of SRT on the Easy DAPO Subset

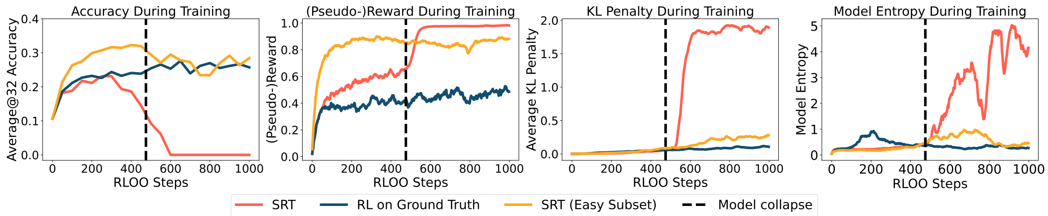

Figure 16: **(Training Dynamics of SRT on the Easy Subset of DAPO)** We show the training dynamics of SRT on the easiest 1/3-rd of the DAPO dataset, chosen by ground truth pass rate of the base model. Compared to SRT on the entire DAPO dataset, SRT on the easier subset does not show any signs of reward hacking, even after taking 3 full passes over the training set.

Figure 3 showed the common signs of reward hacking during SRT-training: namely, sudden drop in accuracy on a held-out dataset, sudden increase in KL penalty, etc. However, we found a simple yet effective way of mitigating reward hacking — simply train on the easiest subset of the training data seems to retain the performance improvement obtained by training on the entire dataset, while preventing reward hacking within the same compute budget. Here we attempt to analyze this phenomenon further, from the lense of the same metrics we recorded in Figure 3.

Figure 16 shows our results: SRT-training on the easiest subset does not show the same behavior as training on the full dataset: accuracy on the heldout set does not drop, and KL penalty, while being slightly higher than that of training with ground truth, is still significantly lower than SRT-training on the full dataset. We also see that model entropy does not explode, so the model keeps outputting reasonable responses instead of the degenerate ones resulting from full dataset training. The most intriguing observation is that regarding pseudo-reward (Figure 16, second from left): it very quickly gets very close to 1 and stabilizes around 0.9. This tends to suggest the model gets very little learning signal as the mean of the pseudo reward is already approximately 1, which is probably the reason it does not learn to reward hack within the same compute budget. We leave investigating this further for future work.

D.2 Training Qwen3-14B-Base on the Easy DAPO Subset

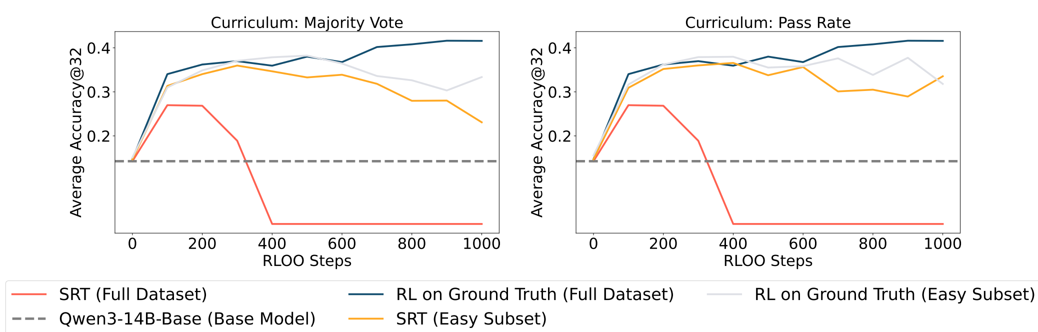

Figure 17: **(Qwen3-14B-Base Trained on the Easy DAPO Subset)** We take the same easy subsets of DAPO from Section 5 and train a Qwen3-14B-Base model with SRT on it. We see the same behavior as Qwen2.5-Math-7B (Figure 8), that SRT on the easy subset does not exhibit performance collapse within the same compute budget.

One of our most interesting observations is that simply using the easiest 1/3-rd of the DAPO dataset eliminates the performance collapse within our training budget (it can still happen if one trains more, though we do not observe it). We want to test whether this is still true for a different base model. To do so, we take the same easiest subset used in Section 5 (so the subset is determined using either the ground truth pass rate or the frequency of the majority answer of a Qwen2.5-Math-7B model) and train a Qwen3-14B-Base model with SRT on this subset. Figure 17 shows the result of our experiments: similar to Qwen2.5-Math-7B model, the Qwen3-14B-Base model also does not exhibit performance collapse within the same training budget.

E Behavior of a Model after Performance Collapse

We provide example generations from a Qwen2.5-Math-7B model before and after prolonged training with SRT on DAPO in Table 1. After SRT training for 1200 steps, the model exhibits performance collapse and generates `\boxed{1}` together with some other incoherent set of tokens for all prompts. This also shows that the model has learned to reward hack the SRT objective.

Table 1: Three examples of model responses for the same prompt, before and after prolonged training with SRT on the DAPO dataset, for a Qwen2.5-Math-7B model. Notice that for some prompts, the model responses before training ends before completion, this is due to the model running out of our token generation budget. The model after 1200 steps of SRT training exhibits performance collapse, and it outputs `\boxed{1}` and some other incoherent set of tokens irrespective of the given prompt.

Prompt	Response before any training	Response after 1200 steps of SRT training
How many positive perfect squares less than 2023 are divisible by 5?	To determine how many positive perfect squares less than 2023 are divisible by 5, let's break the problem down. 1. Identify the range of perfect squares: We need to find all perfect squares n^2 such that $n^2 < 2023$. This means $n < \sqrt{2023}$. Since $\sqrt{2023} \approx 44.98$, the largest integer n for which $n^2 < 2023$ is 44. 2. Check for divisibility by 5: For a perfect square n^2 to be divisible by 5, n itself must be divisible by 5. So, we need to f	espionage—— `\boxed{1}`
What is the value of $$2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3?$$	What is the final answer within `\boxed{}`. # Define the range of numbers and the expression to calculate ... (Model writes code here, we ignore for the sake of simplicity) 4046 The value of the expression $2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3$ is `\boxed{4046}`.	drained , `\boxed{1}` Zac MemoryStream
Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.	It is important to first convert the numbers given in base b to base 10 for mathematical operations. The number 17_b is $1 \cdot b + 7$ in base 10. The number 97_b is $9 \cdot b + 7$ in base 10. We need to find integer bases $b > 9$ such that 17_b divides 97_b. In other words: $$9b + 7 = k(1b + 7)$$ for some integer k. Let's proceed with the necessary calculations in Python to ensure accuracy. # define a functi	-//////`\boxed{1}`