Can Large Reasoning Models Self-Train?

Anonymous authors
Paper under double-blind review

Abstract

Recent successes of reinforcement learning (RL) in training large reasoning models motivate the question of whether self-training, the process where a model learns from its own judgments, can be sustained within RL. In this work, we study this question using majority voting as a simple self-feedback mechanism. On a comprehensive set of experiments on both synthetic and real reasoning tasks, we find that this basic approach improves not only the model's reasoning performance, but also its capability of generating better quality feedback for the next RL iteration, driving further model improvement. Yet our analysis also reveals a critical limitation of such a self-training paradigm: prolonged RL with self-reward leads to reward hacking where models learn to maximize training (pseudo-)reward, resulting in sudden performance collapse. Together, these results highlight feedback design as the central challenge and call for future research on mechanisms to enable prolonged self-improvement.

1 Introduction

Pre-training on human-curated corpora has endowed language models with broad general-purpose capabilities (Brown et al., 2020; Rae et al., 2022), but the supply of such data is becoming a bottleneck as compute scales rapidly (Hoffmann et al., 2022; Sevilla et al., 2022). Reinforcement learning (RL) (Sutton et al., 1998) with verifiable rewards (RLVR) addresses this limitation by using automatic correctness checks, and has already shown success in reasoning and agentic tasks (DeepSeek-AI et al., 2025; OpenAI et al., 2024). Yet in domains where humans cannot provide ground-truth solutions, external feedback breaks down. In these cases, one potential approach to further improving the model is through **self-improvement** where the model evaluates the correctness of its own outputs and uses this signal to refine future generations (Zelikman et al., 2022; Song et al., 2025b; Huang et al., 2025). If sustained iteratively, where the teacher model itself improves, this process could enable continual progress without human supervision.

Prior work on self-improvement has mainly relied on supervised fine-tuning (SFT) (Zelikman et al., 2022; Huang et al., 2023) or direct preference optimization (DPO) (Rafailov et al., 2024; Prasad et al., 2024), where the self-labeling rule is updated only a handful of times (e.g., 1–10 rounds) and *is generally kept fixed within a training round*. These studies show that self-improvement *can* be effective, but leave open whether it can be sustained over longer horizons. Moreover, they are fundamentally bounded by the verification capabilities of the fixed teacher model used to obtain training supervision. In contrast, RL updates the model continuously, a property that has been critical to its success in training reasoning models with verifiable rewards. This success raises a natural question: can self-improvement leverage the same continuous-update paradigm? To study this question, we investigate the setting in which the self-improvement feedback signal is **updated at every gradient step**, fundamentally altering the dynamics of self-improvement compared to earlier work.

In this RL-based setup, the choice of feedback mechanism is critical. If feedback is always correct (e.g., perfectly verifying mathematical solutions), the procedure reduces to standard RL with ground-truth supervision. However, in practice, self-feedback is imperfect, and its design determines whether self-training is effective or not. As a first step, we study the *simplest possible* reward signal: *majority vote*. Wang et al. (2023a) has empirically demonstrated that majority vote tends to have higher accuracy compared to individual generations. Here, we cast the majority vote mechanism as a reward function — granting a positive reward to model outputs that match the most common answer. At the time of the initial publication of our pre-print on arXiv, we were among the first to consider self-training via majority voting pseudo-labels, and we discuss

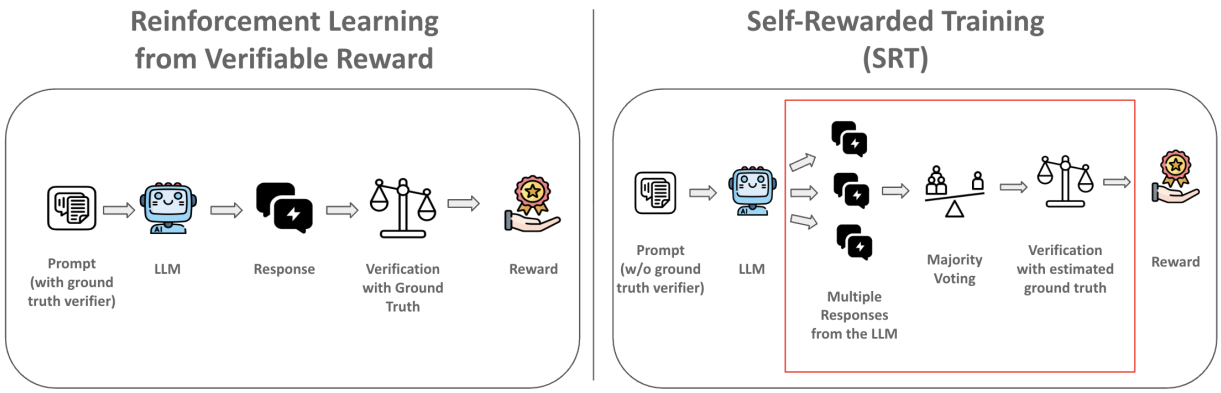


Figure 1: **(Overview of SRT)** In RLVR, one produces the reward for RL training using a ground truth verifier. Contrary to that, SRT does not assume access to a ground truth verifier. Instead it uses majority voting from the model's own generations to estimate the ground truth and use this proxy reward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

concurrent and other related literature in Section 6. Previous work (Huang et al., 2023; Prasad et al., 2024) has employed majority voting mainly as a mechanism to extract better quality generations from a *fixed* teacher policy to then distill it into the student model. At the time of this manuscript's initial posting on arXiv, Zuo et al. (2025) concurrently examined a similar RL procedure with majority voting, but in a different setting, as their focus is on training and testing on the same set of prompts. In contrast, our aim is to use this simple pseudo-label generation mechanism to investigate the validity of RL powered self-training frameworks.

Our comprehensive set of experiments demonstrates that this simple mechanism yields measurable gains over the base model on key reasoning metrics such as maj@k and avg@k success rates. Remarkably, we observe clear improvement in the label generating policy after each gradient step, and this translates to gains over employing labels from a fixed teacher. Moreover, self-training achieves comparable performance to RLVR on 4 different base models. In synthetic tasks where one can control the difficulty of the training dataset, we observe that a simple curriculum-based self-training approach can enable the model to keep climbing on progressively harder tasks without ground-truth labels. However, prolonged training with this framework consistently teaches models to ignore the prompt entirely and output the same template final answer, which maximizes training reward but leads to a complete collapse of the model. We analyze these dynamics in depth and trace them to the self-reinforcing nature of imperfect feedback. These findings identify feedback design as the key challenge that future research should address to sustain self-improvement.

In this work, our contributions are threefold:

(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 show that even simple feedback like majority vote can drive measurable gains in reasoning benchmarks, while **simultaneously improving the supervision signal itself**.

(3) We also analyze one fatal limitation of training 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. We carefully analyze approaches to mitigate reward hacking, laying the groundwork for effective future approaches to sustaining continual model improvement.

2 Preliminaries

Let π_θ denote a language model parameterized by 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 expected *pass rate*, defined as the average accuracy across a distribution of prompts $\mathcal{X}$:

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

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 average 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), RLOO (Ahmadian et al., 2024), Dr. GRPO (Liu et al., 2025), GSPO (Zheng et al., 2025), etc. Implicit to the training method used in our work is the notion of a generation-verification gap (Song et al., 2025b), where generating correct solutions is hard, but verifying them is easy. We present its definition in Appendix A.

3 Self-Rewarded Training

Our objective in this work is to investigate whether *reliable* training supervision for language models can be generated without external labels. The typical practice for online RL involves generating multiple responses to a prompt, then assigning high or low rewards to each generation according to a ground-truth verifier. In the absence of such a verifier, one might develop a mechanism to derive proxy labels. This mechanism then provides a simple recipe for framing self-improvement as an RL problem. At a high level, each iteration proceeds as follows: **(1)** Sample a mini-batch of prompts, **(2)** Determine pseudo-labels, y_{pseudo}, using the mechanism for each prompt, **(3)** Generate n responses per prompt and use agreement with the derived psuedo-labels as an intrinsic binary reward:

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

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

Self-supervision via majority voting. Among several possible choices for determining reasonably accurate pseudo-labels, we explore *majority voting*, since this constitutes the simplest possible self-supervision mechanism. Majority voting has been empirically demonstrated to have higher accuracy compared to individual model generations (Wang et al., 2023a) and is thus a suitable choice to exploit an LLM's inherent generation-verification gap (see Appendix Figure 11) . In our setting, models typically produce

a step-by-step chain of thought followed by a final answer (in the case of regular RL training, this final answer is extracted and matched with the ground truth to produce rewards), so one can group all responses by their final answer to determine the majority vote. Concretely, assume we want to generate pseudo-labels using policy π_{label} (which can be any reasonable policy). This procedure then involves: **(1)** sampling multiple answers per prompt using policy π_{label}, **(2)** grouping answers according to their parsed final solutions, **(3)** estimating the ground truth answer with the most common solution.

Self-Rewarded Training (SRT). The general procedure is described in Algorithm 1, which henceforth shall be called Self-Rewarded Training (SRT) in this paper. Since the method prescribes a specific form of the reward function using model self-consistency, it is compatible with all the common RL training algorithms such as PPO, RLOO, REINFORCE, GRPO, etc. We study the quality of generated labels during training by controlling π_{label}: setting π_{label} to be the base model recovers our familar setting of learning the majority voting decisions of a fixed model (while still using the current policy's rollouts for RL training), and setting π_{label} to be the current policy π_θ after each gradient step allows us to study whether the quality of the learning signal can be concurrently improved during RL training. In the case where we use the current policy π_θ to generate our pseudo-labels, we can reuse them for performing the RL gradient step as well. As the number of generations per prompt typically falls in the range 16-64 (Yu et al., 2025), this variant of SRT incurs no additional compute cost compared to the versions of these algorithms employing ground truth labels. In our work, whether the final answer is correctly formatted and parseable is used to filter responses, but given more compute, one can theoretically employ more sophisticated systems like LLM-as-a-judge (Zheng et al., 2023; Gu et al., 2025) or generative verifiers (Zhang et al., 2025a) to further improve the quality of the training signal. We leave these for future work.

Algorithm 1: Self-Rewarded Training (SRT)

Input: Prompt dataset $\mathcal{X}$
foreach *RL iteration* **do**
  / Inference step */*
  Sample minibatch $\mathcal{B} \subseteq \mathcal{X}$
  foreach *prompt* $x \in \mathcal{B}$ **do**
    Generate n solutions $y^{(1)}, ..., y^{(n)} \sim \pi_{\text{label}}(\cdot|x)$
    Identify majority-vote answer:

$$y_{\text{majority}} \leftarrow \arg\max_{y'} \sum_{i=1}^{n} \mathbf{1}[\text{answer}(y^{(i)}) = y']$$

    Define reward function:

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

  / Gradient update step */*
  Perform RL gradient update using $r(\cdot)$

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 empirical observations in the following section.

4 Experiments

In this section, we present the results of our empirical study. Our primary aim is to answer the two following research questions: **(1)** Can SRT improve an LLMs' reasoning abilities beyond the base model's capabilities, in terms of both the quality of training reward signal and pass@1 performance? **(2)** If yes, can this improvement be sustained indefinitely? We systematically design experiments to answer the questions below.

4.1 Can SRT go beyond the base model's capabilities?

There are two potential axes of improvements over the base model for SRT that can be studied: the improvement in accuracy over held-out prompts, and the improvement in the quality of generated labels themselves during the training procedure. Unlike previous works (Huang et al., 2023; Prasad et al., 2024) that distill the majority voting decision of a fixed policy (typically the base model) into the current model,

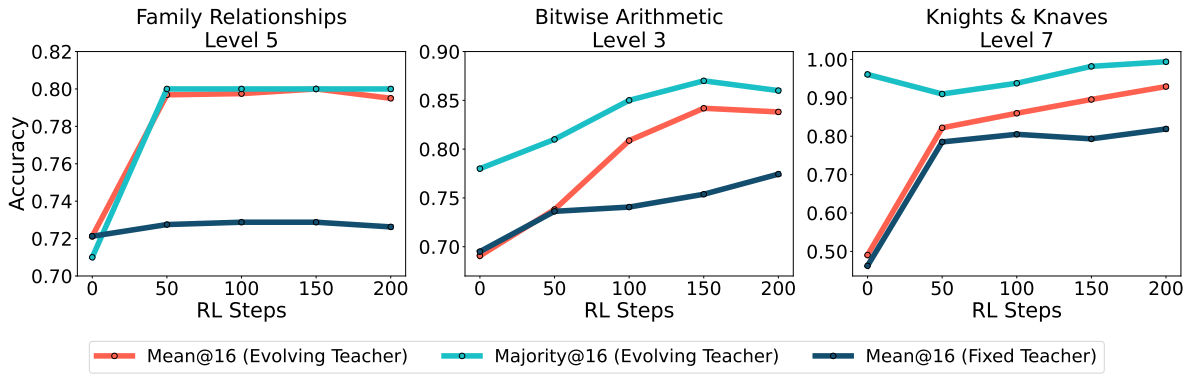


Figure 2: (**SRT improves both performance and quality of generated labels during training.**) We investigate self-training under controlled settings on synthetic reasoning tasks from Reasoning Gym. Remarkably, SRT improves not only the mean accuracy, but the majority voting accuracy as well, which is the source of our training supervision. Improvement in the quality of training signal drives further improvement in performance, as SRT outperforms its variant employing the majority votes from a fixed teacher (base model) as proxy labels.

we want to study **whether the quality of the majority votes of the evolving policy improves** as a result of self-improvement.

Experiments on synthetic reasoning tasks. Complex reasoning tasks like math require domain-specific pre-training/midtraining for online RL to be effective (Wu et al., 2025a; Gandhi et al., 2025); it is also more difficult to control the difficulty of the individual tasks that the model sees during training. Therefore, we first study this question using synthetic reasoning tasks from REASONING GYM (Stojanovski et al., 2025), a collection of over 100 reasoning environments for reinforcement learning with verifiable rewards. These tasks can also be generated with adjustable difficulty, making it a suitable test-bed for our work. Concretely, we use 3 tasks from Reasoning Gym: **(1) Family Relationships**, a logic puzzle involving a group of individuals connected via different relationships, and the model has to reason about the relationship between two individuals within this group, **(2) Bitwise Arithmetic**, a task for testing models' understanding of Bitwise Arithmetic operations, and **(3) Knights & Knaves**, a logic puzzle involving characters who always either tell the truth (knights) or always lie (knaves), and the challenge is to deduce who is who based on their statements. Appendix K shows example prompts from each of these tasks. We use a Qwen-3-4B-Base (Yang et al., 2025) model for all our reasoning gym experiments.

Since our goal is to study whether SRT can improve performance on top of a reasonably strong base model, following the setting of Lee et al. (2025), we first train the base model with GRPO using ground truth labels on the easiest difficulty setting. This is used to teach the base model proper formatting rules and how to solve the basic task before we train using SRT on the next level of difficulty without ground-truth labels. The detailed training settings can be found in Appendix B.3.

Results. Figure 2 shows our main results: SRT using the current policy as the label generator improves **both** avg@16 and majority vote@16 accuracy (the supervision signal) on all 3 reasoning gym tasks. Since we derive our training signal from the majority votes of the current policy evolving with every RL step — this demonstrates that self-improvement using SRT can progressively improve the **quality of the pseudo-labels as well**. Note that this would not be possible in prior works (Prasad et al., 2024; Huang et al., 2023) which distills the majority votes of a *fixed teacher* policy throughout one or a few rounds of training. We expect the improvement in the *evolving teacher* policy to result in further performance gain. To validate this, we compare SRT with a variant of the same algorithm where we use the majority votes of the fixed starting policy instead of the evolving current policy as pseudo-labels. Figure 2 shows their comparison: in Family Relationships and Bitwise Arithmetic, we see larger gains in majority voting performance, and likewise SRT outperforms its fixed teacher variant substantially, by 10% for Bitwise Arighmatics, 8% in Family Relationship, and 6% in Knights and Knaves. On Knights & Knaves, the starting policy already has >90% majority voting accuracy, and we see the difference between the evolving teacher and fixed teacher variants of SRT to be smaller

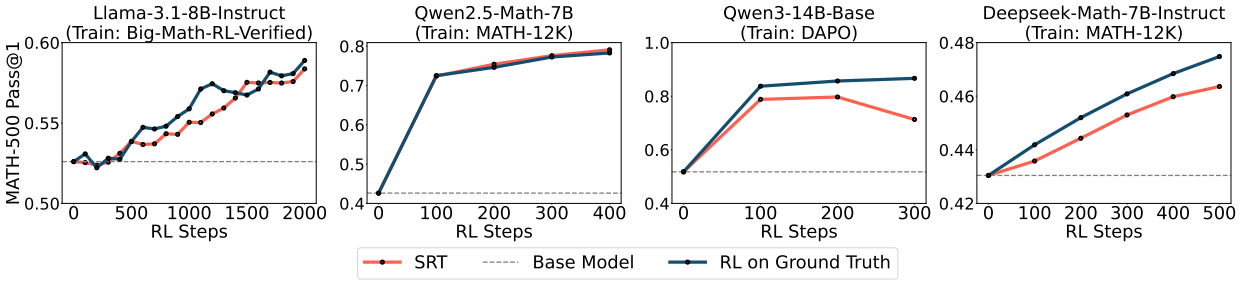


Figure 3: **(Evaluating SRT on real-world math problems.)** Comparison between SRT and RL with ground truth across different base models and training datasets. Following Oertell et al. (2024), all models are trained using RLOO (for experiments with GRPO, see Figure 8) and tested using average pass@1 accuracy on MATH-500. SRT achieves comparable performance to that of ground-truth training across different base models. For training curves using more combinations of (train, test) dataset pairs, refer to Appendix C.1 and C.2.

here. Furthermore, these performance gains cannot be explained by learning to format properly: since the model was trained on the easiest difficulty level, it can already format its responses correctly in most cases (Appendix E), and we see performance climbing after saturation in the model's format correctness. Overall, on the synthetic reasoning tasks, SRT clearly pushes the model beyond its starting capabilities, showing the promise of self-improvement even from this basic recipe for self-supervision.

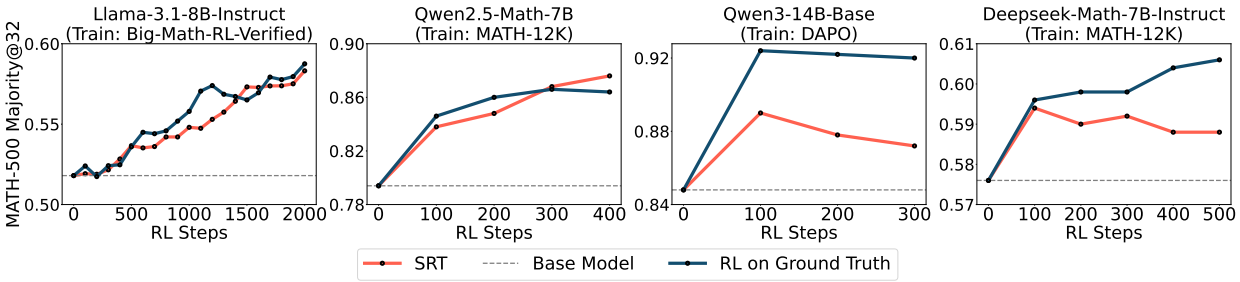


Figure 4: **(Majority@32 accuracy comparison between SRT and RL with ground truth)** We compare the majority@32 accuracy, as opposed to average accuracy shown in Figure 3. **Note that for Llama-3.1-8B-Instruct, we use the official model card evaluation temperature of 0, hence majority@32 is the same as average@32 accuracy.** SRT shows improvement in the quality of the majority votes themselves, which distinguishes our algorithm from that of learning from a fixed teacher's majority votes.

Real-world reasoning tasks. Armed with these insights, we next test our algorithm on real-world reasoning tasks in the math domain (for other tasks like coding, it is unclear how to group generations to find the majority vote). To investigate the generality of our algorithm, we run a comprehensive set of experiments with 4 different base models of different sizes: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-Math-7B (Yang et al., 2024), Qwen3-14B-Base (Yang et al., 2025), and Deepseek-Math-7B-Instruct (Shao et al., 2024). We also use a comprehensive suite of train and test datasets: namely, for training we employ Big-Math-RL-Verified (Albalak et al., 2025), MATH-12K (Hendrycks et al., 2021), and DAPO (Yu et al., 2025); for evaluation, we use MATH-500 (Hendrycks et al., 2021), AIME 2024, AIME 2025 and AMC. We also experiment with different RL algorithms such as GRPO and RLOO, refer to Appendix B.3 and Figure 8 for more details (they show no noticeable difference in behavior). All training and validation runs used a temperature value of 1.0, except for Llama-3.1-8B-Instruct, where we evaluated using the official model card setting of temperature 0 (greedy decoding). Note that different sampling temperatures in validation can greatly affect the base model performance, a phenomenon we explore in Appendix D. Since our goal is to show that SRT and RL with ground truth lead to a similar improvement over the base model and we are not concerned about the absolute improvement, our conclusions hold regardless of the decoding temperature.

We show a summary of our empirical findings using different base models and training datasets in Figures 3 and 4 (for training curves using more combinations of (train, test) dataset pairs, refer to Appendix C.1

and C.2). On all instances, SRT improves both avg@16 and majority vote@16 accuracies on heldout MATH-500 prompts, and performs on par with regular RL training with ground truth verification. More impressively, the observations hold for base models like Llama-3.1-8B-Instruct, which is known to be particularly difficult for RL training on reasoning tasks (Gandhi et al., 2025), improving its average accuracy from 52.6% to nearly 60%.

We also compare SRT with its offline variants: SFT on the majority vote (Huang et al., 2023), DPO and ScPO (Prasad et al., 2024) employing contrastive learning between the majority vote and non-majority vote answer in Table 2. We observe that SRT retains better performance compared to its offline variants distilling the majority vote decisions of the base policy, **showing the benefit of self-improvement in the label-generating policy**. For more details about the baselines, please refer to Appendix F.

Takeaway 1: SRT can improve reasoning capabilities beyond the base model

On both synthetic and real reasoning tasks, SRT improves average and majority voting accuracies, showing ability gains beyond the base model. Specially, improvement in majority voting accuracy also signifies improvement of the quality of self-supervision during training, demonstrating a promising path forward to self-improvement.

4.2 Can Self-Improvement from SRT be Sustained Indefinitely?

Given the strong performance of SRT in various reasoning tasks and model architectures, an important question is whether self-training can be maintained over extended iterations. Similar to the prior section, we first test SRT on synthetic tasks with controllable difficulty to rigorously study its properties, and then test the resulting insights on real-world reasoning domains.

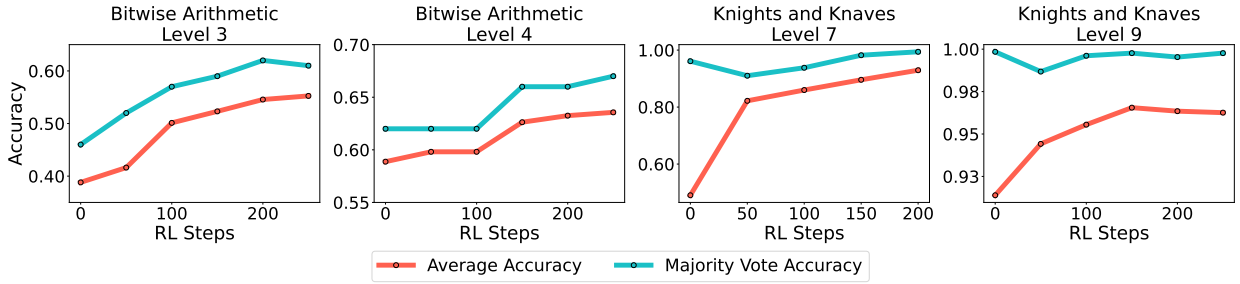


Figure 5: **(Multi-level climbing on Reasoning Gym using curriculum)** The Qwen3-4B-Base model can climb on progressively more difficult tasks without ground truth labels via a simple curriculum strategy — where we train an earlier level's final checkpoint with SRT on the next difficulty level. This approach also seems to improve both average and majority voting accuracy on each level.

Multi-level self-improvement on synthetic tasks using curriculum. Since Reasoning Gym provides a built-in way to control the difficulty of generated tasks, we first investigate whether self-training on an easier set of tasks can produce a model capable of self-improvement on progressively harder levels of difficulty. To do so, we choose 2 Reasoning Gym tasks: Bitwise Arithmetic and Knights & Knaves. Similar to our previous setting, we first train using RL on ground truth labels on the easiest level of difficulty, then progressively train on harder levels without ground truth labels (i.e., SRT on level 5 starts with the checkpoint obtained from SRT on level 4, and so on). For more details, refer to Appendix B.3.

Figure 5 shows our primary results: in this controlled setting, SRT is able to maintain self-improvement on progressively harder difficulties. In particular, SRT can show reasonable improvement in Bitwise Arithmetic Level 4 after being initialized on Level 2 with ground-truth training, and also progressively climb to near 100% accuracy on Knights & Knave Level 9 after being trained with ground-truth on Level 2 only (intermediate levels are trained with SRT).

Extended SRT-training on math problems. Next, we test our insights on real-world math problems. Specifically, we take take the same 4 base models as in the previous section, and train them on a difficult

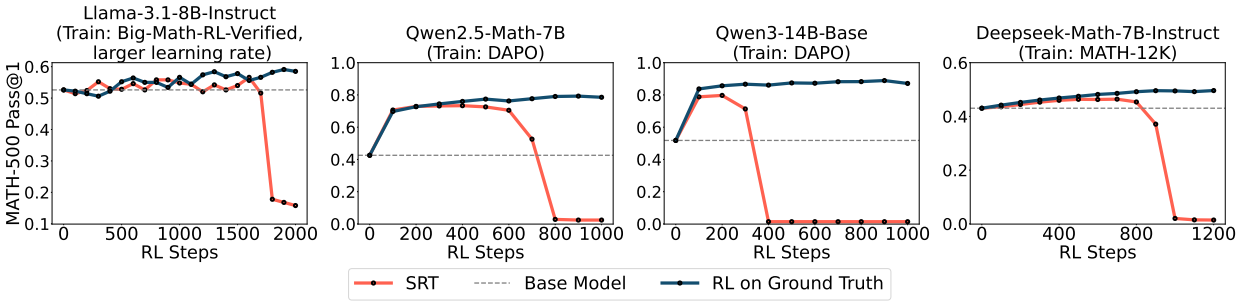


Figure 6: **(Extended self-training leads to model collapse)** Inspired by multi-level improvement on reasoning gym tasks, we take four LLMs with strong math abilities from pretraining, and train them with SRT for an extended period of time. SRT improves performance at first, but then demonstrates complete model collapse on all 4 base models. (Note: on Llama-3.1-8B-Instruct, the learning rate used in Figures 3 and 4, 10^{-7}, does not lead to model collapse within our training budget, but 3×10^{-7}, a slightly higher learning rate does — we hypothesize that with an extended training run, even 10^{-7} would lead to model collapse. For more details on the effect of hyperparameters on model collapse, refer to Appendix G.)

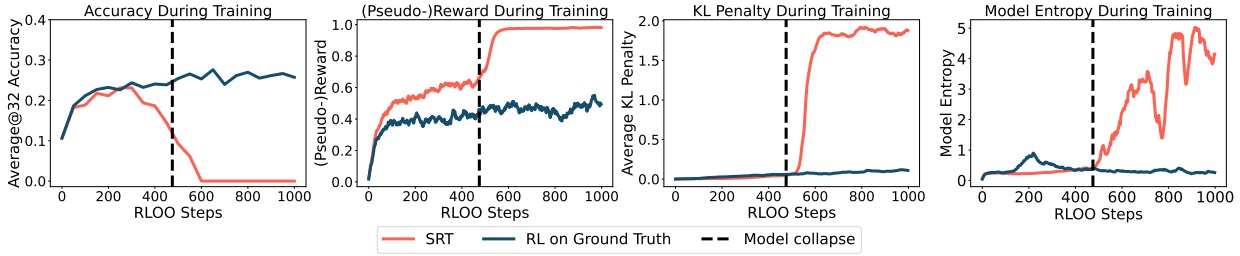


Figure 7: **(Self-Training Dynamics)** Extended training via SRT can lead to reward hacking, as demonstrated by the sudden hike in KL penalty and training (psuedo-)reward, but collapse of accuracy on the held-out test sets.

math dataset through an extended number of iterations using SRT. Figure 6 demonstrates our surprising finding: while SRT initially increases the base model's performance at a comparable rate with ground-truth RL training, extended training using SRT leads to sudden performance collapse. **We observe performance collapse or degradation from extended SRT-training across all models and training datasets,** and record these in detail in Appendix C.1 and C.2. Given that this surprising phenomenon deserves more investigation, we study it in more detail next.

What happens after SRT peaks in performance? To develop a clearer understanding of the underlying reasons for this phenomenon, in this section we investigate this SRT-induced model collapse closely.

We plot the training statistics of the SRT objective (Eqn. 6) in Figure 7. 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 Table 1) confirms this hypothesis: **after collapse, the model outputs a very high entropy, essentially random, set of tokens followed by the same "template" final answer** (for example, `\boxed{1}`) 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. This behavior is also related to the well-known simplicity bias in neural networks (Palma et al., 2019; Valle-Pérez et al., 2019; Mingard et al., 2020; 2025), as

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 ...	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) 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

Table 1: Two 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.

well as the Occam's razor, where neural networks tend to find the simplest solution that generalizes to the observed signal — in this case this leads to the same final template answer for all prompts.

Takeaway 2: Self-Training benefits may not extend indefinitely

The question of whether self-training can be extended indefinitely has mixed results: while under controllable difficulty, SRT can keep improving beyond the base model on progressively more difficult tasks, training on real-world math problems demonstrate the phenomenon of reward hacking — sustained self-improvement requires developing additional regularization measures to be effective.

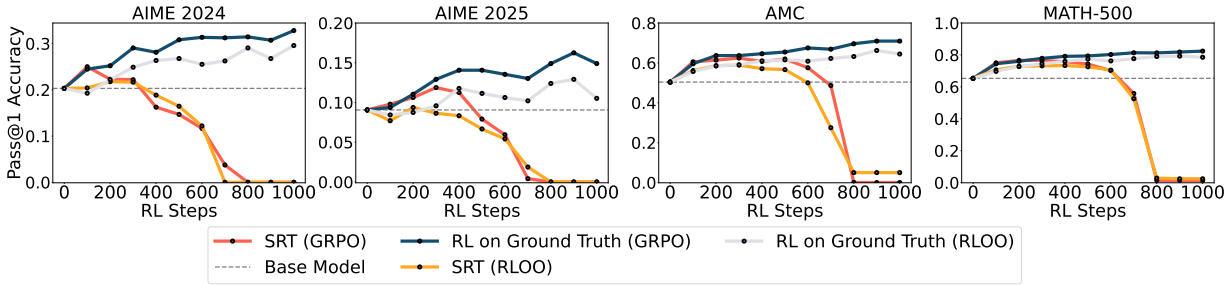


Figure 8: **(GRPO vs RLOO comparison)** We compare the behavior of SRT under two different RL optimization algorithm: GRPO vs RLOO. All experiments use a Qwen2.5-Math-7B model trained on DAPO, with the other hyperparameters being the default ones described in Appendix B.3. While SRT with GRPO seems to achieve higher performance than that of SRT employing RLOO, ultimately prolonged training using both algorithms lead to reward hacking and model collapse on all test datasets.

Additional experiments and ablations. We have run additional experiments and ablations to study the behavior of self-training under different training scenarios. The main conclusions are as follows: **(1)** Choice of RL algorithm (GRPO vs RLOO) does not affect the final outcome of SRT-training (Figure 8) , **(2)** increasing KL coefficient to incentivize the model to stay close to the base policy also does not mitigate reward hacking, as the training signal from the reward hacked solution is too strong (Figure 20), **(3)** decreasing learning rate seems to delay model collapse but not eliminate it, and we hypothesize that prolonged training with lower

learning rates would still result in complete model collapse (Figure 21), and **(4)** Surprisingly, reducing the number of generations per prompt injects noise into the training signal, which delays quick model collapse by exploiting the majority voting answer (Figure 22). Additional experiments related to SRT training, including training curves on all test datasets and the effect of tuning additional hyperparameters like entropy coefficient, can be found in Appendix C and G.

5 Can We Prevent Model Collapse in Self Training?

As discussed before, 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) A *data-driven* curriculum-based strategy to enhance model performance beyond simple early stopping.

5.1 Early Stopping

In our algorithm, even a small labeled validation dataset can effectively identify the peak performance point during self-training, thereby mitigating model collapse. Figure 9 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. The vertical line in Figure 9 marks early stopping using only 1% of DAPO as validation, with performance on other datasets remaining near-optimal.

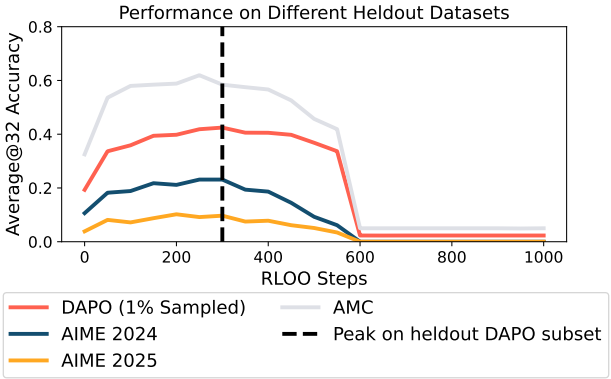


Figure 9: **(Early Stopping is Effective)** The peak performance occur at nearly the same point for all heldout sets, so using any would be effective for early stopping.

5.2 Self-Training with Curriculum Learning

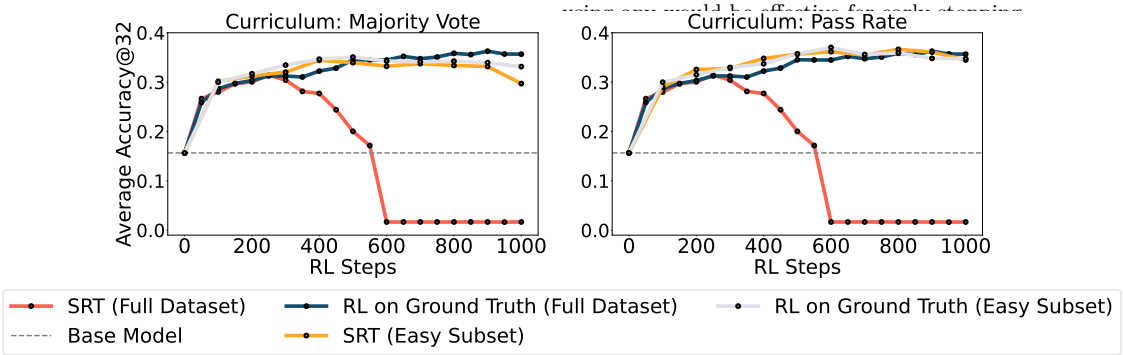


Figure 10: **(Curriculum-Based Self-Training)** Performance of SRT on curated subsets containing the easiest 1/3 of prompts from the DAPO dataset, selected based either on model pass rate or frequency of the majority vote. Training on these easier subsets prevents reward hacking even after extensive training (3 epochs), demonstrating the effectiveness of curriculum learning strategies in sustaining continual model improvement.

Our third approach, curriculum learning, is motivated by the observation that 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 pretrained 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., 2025b; 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 10 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. More importantly, **we did not observe training collapse even after 3 epochs of training on 1/3rd of "easy" DAPO** dataset. These promising results suggest that curriculum strategies may further extend the benefits of SRT, which we leave as a future research direction. Additional experiments related to curriculum learning can be found in Appendix I.

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., 2024a) 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., 2025b) 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., 2025a), 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. Previous works have studied self-improvement by generating labels through majority voting (Prasad et al., 2024; Huang et al., 2023), but these works are typically confined to one or a few rounds of SFT or DPO (Rafailov et al., 2024). This essentially involves distilling majority voting labels from a fixed policy into the current policy over each round of training. In contrast, we explore *online RLVR's* potential in self-improving LLMs where the label generating policy evolves after every gradient step. A few concurrent works (Chen et al., 2025; Prabhudesai et al., 2025; Zhao et al., 2025b; Shao et al., 2025) have also explored various forms of self-rewarding mechanisms through majority voting or a similar metric of self-consistency (e.g., token/sequence level entropy) in Online RLVR. Finally, in a recent work, (Song et al., 2025b) 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. We refer the interested reader to Gao et al. (2025) for a survey of other works associated with self-evolving LLM agents.

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., 2025b; 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. Finally, a concurrent and complementary work, Zhang et al. (2025b), has also demonstrated the failure of various self-reward mechanisms to sustain self-improvement under prolonged training, which further validates the observations in this work.

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 H.1 as an extension of SRT.

7 Limitations and Conclusion

In this work, we examine a simple strategy of leveraging an LLM's self-consistency to train it via an RL framework. Our experiments on synthetic reasoning tasks with controllable difficulty levels demonstrate the promise of such a framework: not only can LLMs improve their performance on these tasks, but they can also improve the quality of their self-supervision for the next training step. Our analysis, however, also reveals the limitation of leveraging self-consistency as training reward: prolonged training under such a framework can lead to reward hacking and complete model collapse. Future investigations could explore how to develop more robust forms of verification, extend self-verification to other domains like coding, and curriculum learning strategies that expose the model to problems of only the right difficulty. Additionally, leveraging LLM-as-judges or generative verifiers to improve the training signal, or employing additional consistency regularization between the chain-of-thought and the final answer to mitigate reward hacking can be promising future research. We leave these and other promising directions for sustained self-improvement for future work.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go MAD. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ShjMHfmPs0.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

Quentin Bertrand, Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. 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.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models, 2025. URL https://arxiv.org/abs/2508.03682.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 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.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective on reinforcement learning for llms, 2025. URL https://arxiv.org/abs/2506.14758.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, Ethan Perez, and Evan Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models, 2024. URL https://arxiv.org/abs/2406.10162.

Yunzhen Feng, Elvis Dohmatob, Pu Yang, Francois Charton, and Julia Kempe. 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.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

Huanang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence, 2025. URL https://arxiv.org/abs/2507.21046.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Tomasz Korbak, Henry Sleight, Rajashree Agrawal, John Hughes, Dhruv Bhandarkar Pai, Andrey Gromov, Dan Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. 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.

Nate Gillman, Michael Freeman, Daksh Aggarwal, Chia-Hong Hsu, Calvin Luo, Yonglong Tian, and Chen Sun. Self-correcting self-consuming loops for generative model training, 2024. URL https://arxiv.org/abs/2402.07087.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.15594.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. Glore: When, where, and how to improve llm reasoning via global and local refinements, 2024. URL https://arxiv.org/abs/2402.10963.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Jian Hu, Jason Klein Liu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025. URL https://arxiv.org/abs/2501.03262.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WJaUkwci9o.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (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/.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 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.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *DeepRLStructPred@ICLR*, 2019. URL https://api.semanticscholar.org/CorpusID:198489118.

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, 2025. URL https://arxiv.org/abs/2411.15124.

Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges, 2025. URL https://arxiv.org/abs/2502.01612.

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, 2025. URL https://arxiv.org/abs/2503.20783.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A. Louis. Neural networks are a priori biased towards boolean functions with low entropy, 2020. URL https://arxiv.org/abs/1909.11522.

Chris Mingard, Henry Rees, Guillermo Valle-Pérez, and Ard A. Louis. Deep neural networks have an inbuilt occam's razor. *Nature Communications*, 16:220, 2025. doi: 10.1038/s41467-024-54813-x. URL https://doi.org/10.1038/s41467-024-54813-x.

Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. Learning math reasoning from self-sampled correct and partially-correct solutions. In *ICLR*, 2023. URL https://openreview.net/forum?id=4D4TSJE6-K.

Owen Oertell, Wenhao Zhan, Gokul Swamy, Zhiwei Steven Wu, Kiante Brantley, Jason Lee, and Wen Sun. Heuristics considered harmful: Rl with random rewards should not make llms reason, 2024. Preprint.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, et al. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Giacomo De Palma, Bobak Toussi Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions, 2019. URL https://arxiv.org/abs/1812.10156.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. Language model self-improvement by reinforcement learning contemplation. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=38E4yUbrgr.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization, 2024b. URL https://arxiv.org/abs/2404.19733.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning, 2025. URL https://arxiv.org/abs/2505.22660.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. Self-consistency preference optimization, 2024. URL https://arxiv.org/abs/2411.04109.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, et al. Scaling language models: Methods, analysis & insights from training gopher, 2022. URL https://arxiv.org/abs/2112.11446.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional unalignment: Likelihood displacement in direct preference optimization, 2025. URL https://arxiv.org/abs/2410.08847.

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

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

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. 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.

Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning, 2022.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2025. URL https://arxiv.org/abs/2506.10947.

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. URL https://arxiv.org/abs/2402.03300.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget, 2024a. URL https://arxiv.org/abs/2305.17493.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross J. Anderson, and Yarin Gal. 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.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 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.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning, 2025a. URL https://arxiv.org/abs/2509.06941.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=mtJSMcF3ek.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization, 2020. URL https://openreview.net/forum?id=HyezmlBKwr.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 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.

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

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data, 2024. URL https://arxiv.org/abs/2404.14367.

Fahim Tajwar, Yiding Jiang, Abitha Thankaraj, Sumaita Sadia Rahman, J Zico Kolter, Jeff Schneider, and Ruslan Salakhutdinov. Training a generally curious agent, 2025. URL https://arxiv.org/abs/2502.17543.

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl, 2025. URL https://arxiv.org/abs/2506.09477.

Kimi Team, 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, Haiqing Guo, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, 2019. URL https://arxiv.org/abs/1805.08522.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 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.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas 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, 2025. URL https://arxiv.org/abs/2504.20571.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (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/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

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

Haoze Wu, Cheng Wang, Wenshuo Zhao, and Junxian He. Mirage or method? how model-task alignment induces divergent rl conclusions, 2025a. URL https://arxiv.org/abs/2508.21188.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge, 2025b. URL https://openreview.net/forum?id=lbj0i29Z92.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. 2024. URL https://arxiv.org/abs/2410.23123.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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, Weinan Dai, 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, 2025. URL https://arxiv.org/abs/2503.14476.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Edwin Zhang, Vincent Zhu, Naomi Saphra, Anat Kleiman, Benjamin L. Edelman, Milind Tambe, Sham M. Kakade, 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.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025a. URL https://arxiv.org/abs/2408.15240.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for llm reasoning, 2025b. URL https://arxiv.org/abs/2506.17219.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data, 2025a. URL https://arxiv.org/abs/2505.03335.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards, 2025b. URL https://arxiv.org/abs/2505.19590.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation, 2025. URL https://arxiv.org/abs/2509.15194.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Definition of Generation-Verification Gap

The single-generation accuracy is defined as:

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

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

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

We define verification accuracy as:

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

We say that a positive *generation-verification gap* occurs whenever $\mathrm{Acc}_{\mathrm{ver}}(\theta, n) > \mathrm{Acc}_{\mathrm{gen}}(\theta)$. 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

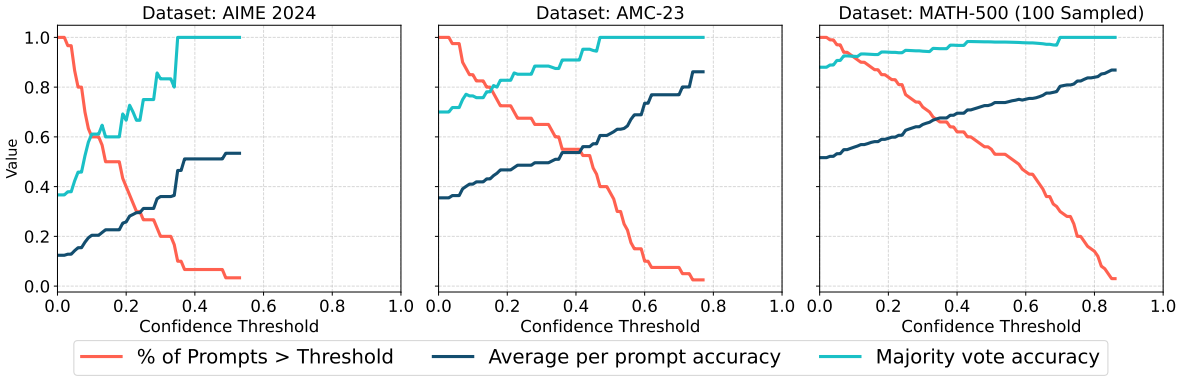


Figure 11: Three of our test datasets display evidence of generation verification gap through majority voting. **The positive gap between majority voting accuracy and per prompt accuracy means LLMs can utilize this gap as a learning signal to improve their average accuracy**. x axis refers to the threshold cut off for 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 RL Algorithms

We use two RL algorithms in this work: RLOO(Ahmadian et al., 2024; Kool et al., 2019) and GRPO (Shao et al., 2024). Recent work (Oertell et al., 2024) has shown that heuristic policy gradient algorithms like GRPO can produce unexpected results by increasing or decreasing reasoning performance even under random rewards, where policy gradient should be zero in expectation, and that RLOO does not have this problem. Since SRT is compatible with both RL algorithms, we experiment with both and observe no noticeable difference in the resulting behavior.

In our implementation (`verl`), we use the following RL objective for both RLOO and GRPO:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\ \{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(w_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}(w_{i,t}(\theta),\ 1-\varepsilon,\ 1+\varepsilon)\hat{A}_{i,t}\Big)\Bigg]$$

where $w_{i,t}(\theta)$ is the importance ratio, defined as:

$$w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x,\ y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x,\ y_{i,<t})}$$

Since we operate fully on-policy, i.e., one RL step per one batch of generated rollouts, this is always one in our experiments. The same advantage defined at a sequence level is applied to each token in the sequence, so henceforth we will drop the t from the notation as well.

The main difference between GRPO and RLOO then stems from their use of different advantage functions. RLOO objective uses the following advantage function:

$$\frac{1}{G}\sum_{i=1}^{G}[R(y_{(i)}, x) - \frac{1}{G-1}\sum_{j\neq k}R(y_{(j)}, x)]$$

whereas GRPO uses the following advantage function:

$$\hat{A}_i = \frac{r(x, y_i) - \text{mean}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}{\text{std}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}$$

Here G is the number of online samples generated. Both RLOO and GRPO 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 for both algorithms.

In our implementation, we did not add KL penalty to the loss function, rather to the reward itself while running RLOO, following recent work such as Tang & Munos (2025). In verl framework, this can be configured using `algorithm.use_kl_in_reward=True` and `actor_rollout_ref.actor.use_kl_loss=False`. However, this does not work for GRPO due to advantage normalization by the standard deviation, and so for GRPO we add KL penalty to the loss function directly. 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` to -1, and `top_p` to 1 for rollouts generated during RL rollouts. Decoding temperature used for validation varies in different settings, see Appendix B.3 and D for more discussion. We cut off maximum prompt length at 1024 and maximum response length to 3072 (note: 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 ∼15000 GPU hours (including preliminary studies and failed runs). All the final results listed in this paper can be replicated within 2000 H200 GPU hours.

B.3 Details on Training Settings

We choose Family Relationships, Bitwise Arithmetic, and Knights & Knaves (Xie et al., 2024) tasks from Reasoning Gym (Stojanovski et al., 2025) for our experiments. Examples for each task is shown in Appendix K.

For the **Bitwise Arithmetic** task, *Level* refers to the `difficulty` parameter. The model was first trained with level 2 data for 950 steps, reaching 97% accuracy. We then used this initialized model to train with SRT on levels 3 and 4.

For the **Family Relationships** task, we trained a model to 99% accuracy on the level 4 dataset. Here, *Level* corresponds to the parameters `min_family_size` and `max_family_size`, both set to 4. We then applied SRT on level 5.

For the **Knights & Knaves** task, we varied only the `n_people` parameter as the difficulty control. We first trained a model with difficulty level 2 to 99% accuracy, and then used that checkpoint to further experiment with SRT, climbing from level 2 to 3, 3 to 5, 5 to 7 and 7 to 9. **We only report levels 7 and 9 in the paper since they are the highest level difficulty among our experiments**.

Across all multi-level experiments, we applied SRT progressively. For example, in Bitwise Arithmetic we trained on level 2 with ground truth supervision; then, starting from the level 2 checkpoint, we applied SRT on level 3; finally, we repeated SRT again on level 4 using the checkpoint from level 3. For comparing against **SRT with fixed teacher**, we use the same starting policy (Qwen3-4B-Base trained with ground-truth on the easiest difficulty level on each task) and generate the same number of rollouts per prompt using temperature 1.0 and perform majority voting among these rollouts to generate our pseudo-labels.

Default training hyperparameters for Reasoning Gym tasks. For all Reasoning Gym experiments, we used the `Qwen3-4B-Base` model, with GRPO as the main algorithm. The learning rate was set to `1e-6` and the KL penalty to `0.0001`. For all experiments, we used 32 rollouts per prompt for training and 16 rollouts for evaluation.

Default training hyperparameters for Math Datasets.

- **Qwen2.5-Math-7B**: Learning rate 10^{-6}, KL penalty coefficient 0.001, decoding temperature for training and evaluation rollouts 1.0, top-p 1.0, and no top-k sampling.
- **Qwen3-14B-Base**: Same default hyperparameter setting as Qwen2.5-Math-7B.
- **Llama-3.1-8B-Instruct**: Learning rate 10^{-7}, KL penalty coefficient 0.001, decoding temperature for training is 1.0 and evaluation is 0.0, top-p 1.0, and no top-k sampling. We subsample the Big-Math-RL-Verified dataset to only keep prompts that has average Llama-3.1-8B-Instruct pass rate between 0.3 and 0.7, since the model is unable to improve during training otherwise.

- **Deepseek-Math-7B-Instruct**: Learning rate 10^{-7}, KL penalty coefficient 0.001, decoding temperature for training is 1.0 and evaluation is 0.7 (following official protocol), top-p 1.0, and no top-k sampling.

C Additional Experimental Results

C.1 Qwen2.5-Math-7B

Here we compare the performance of training Qwen2.5-Math-7B with SRT and RL with ground truth on each individual test set for the sake of completeness. Figures 12, 13, and 14 show the detailed results when we train on DAPO, MATH-12K, and AIME (1983-2023) respectively. Additionally, when training on MATH-12K and DAPO, we also evaluate the intermediate checkpoints on the heldout set MATH-500, which is reported in Figure 13. **Since MATH-500 contains 500 examples, calculating average@32 accuracy becomes expensive, and hence we could not use it as a test set for all our training setups**.

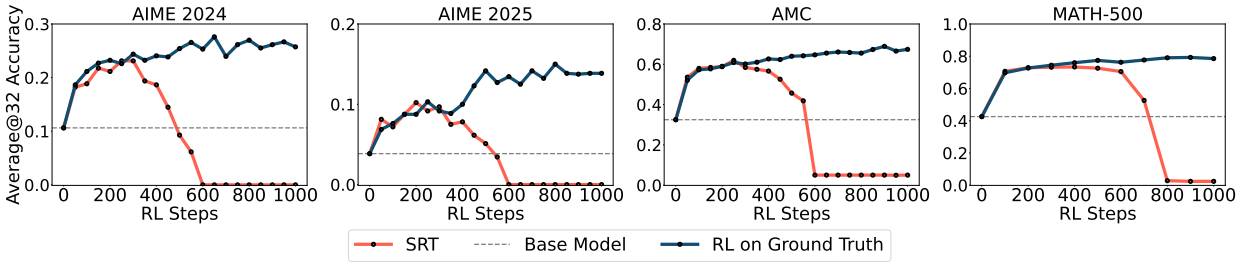


Figure 12: **(Individual test set performance during training on DAPO)** We record the average@32 accuracy during training Qwen2.5-Math-7B on DAPO, on three heldout test sets: AIME 2024, AIME 2025 and AMC. In all three cases, SRT performance collapses, while training with ground truth keeps improving steadily.

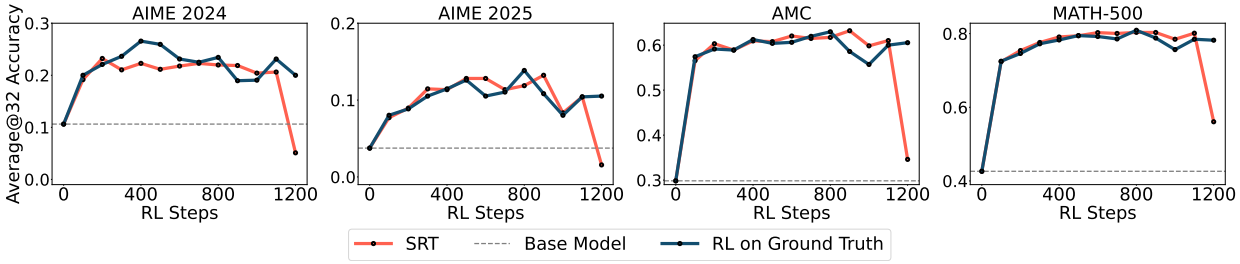


Figure 13: **(Individual test set performance during training on MATH-12K)** We record the average@32 accuracy during training Qwen2.5-Math-7B on MATH-12K, on three heldout test sets: AIME 2024, AIME 2025 and AMC. We also evaluate intermediate checkpoints on MATH-500 since we are training on MATH-12K (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. However, performance collapses after 1200 RL steps, similar to our other observations.

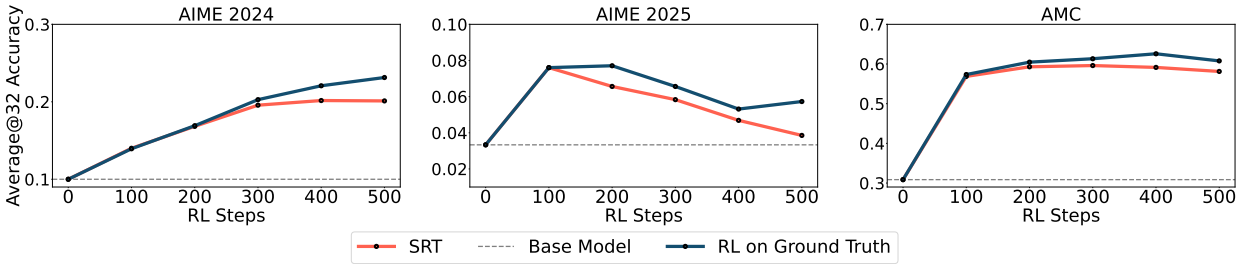


Figure 14: **(Individual test set performance during training on AIME (1983-2023))** We record the average@32 accuracy during training Qwen2.5-Math-7B on AIME (1983-2023), on three heldout test sets: AIME 2024, AIME 2025 and AMC. SRT performs similarly or better compared to training with ground truth labels over 10 epochs of training.

C.2 Qwen3-14B-Base

In addition to Qwen2.5-Math-7B, we apply our algorithm on 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.

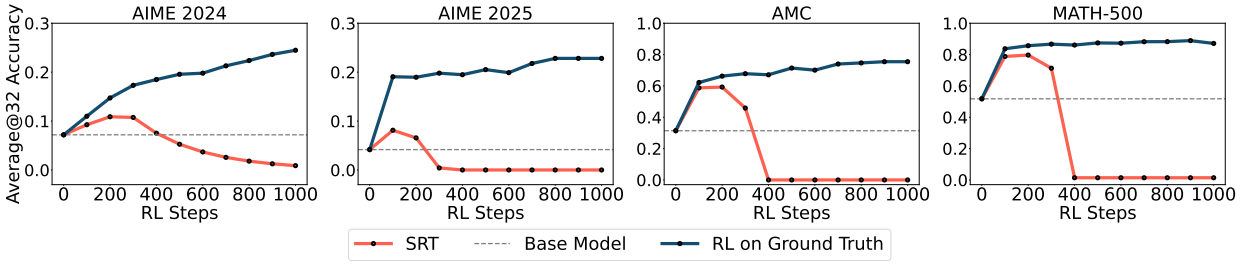


Figure 15: **(Individual test set performance during Qwen3-14B-Base on DAPO)** We record the average@32 accuracy during training a Qwen3-14B-Base model on DAPO, on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

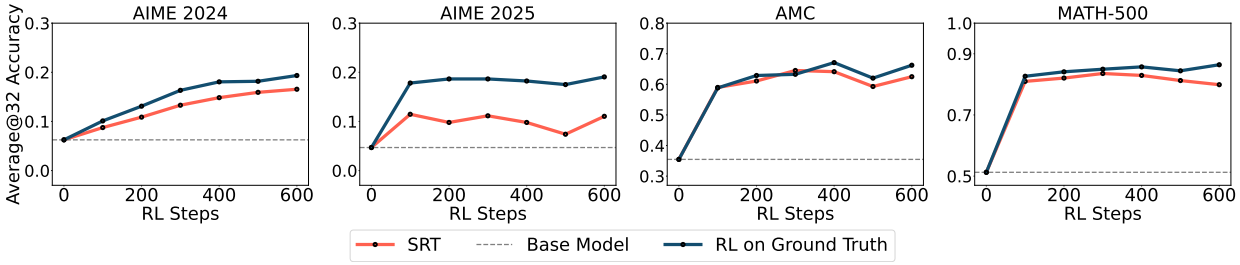


Figure 16: **(Individual test set performance during training Qwen3-14B-Base on MATH-12K)** We record the average@32 accuracy during training a Qwen3-14B-Base model on MATH-12K, on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

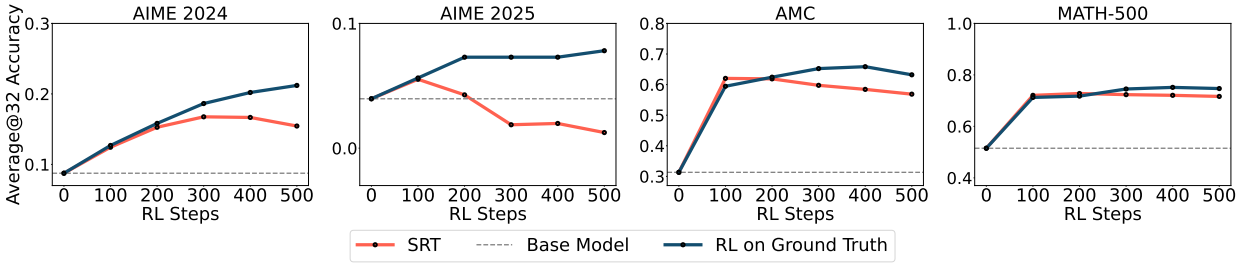
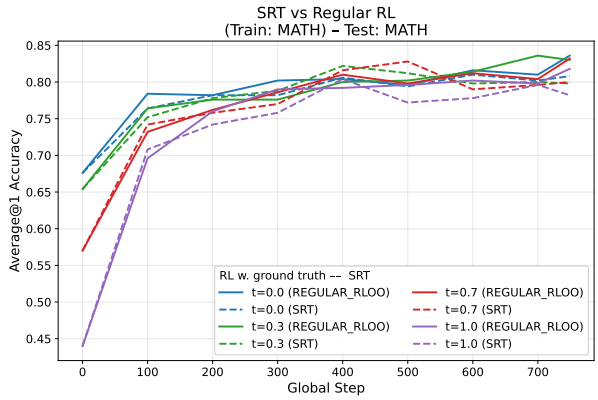


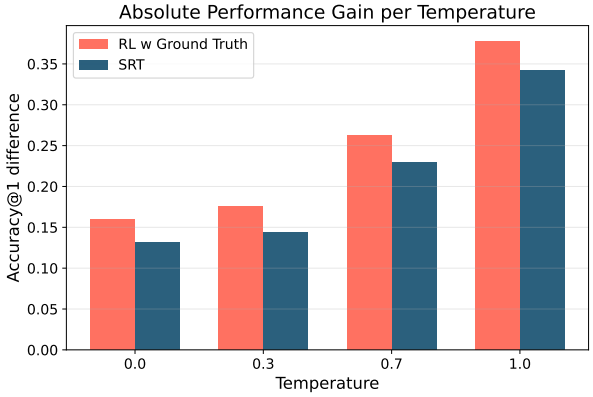
Figure 17: **(Individual test set performance during training Qwen3-14B-Base Model on AIME (1983-2023)** We record the average@32 accuracy during training a Qwen3-14B-Base model on AIME (1983-2023), on four heldout test sets: AIME 2024, AIME 2025, AMC, and MATH-500.

Figures 15, 16, and 17 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 Effect of Decoding Temperature (Qwen2.5-Math-7B)



(a) Math-500 evaluation accuracy during training, when decoded under various temperature (Qwen2.5-Math-7B).



(b) Absolute performance improvement after one epoch of training when decoded under various temperatures (Qwen2.5-Math-7B).

Figure 18: Our method (SRT) performs consistently regardless of decoding temperature for validation (Figure 18a). All experiments are run using Qwen2.5-Math-7B as the base model, trained on MATH-12K and tested on MATH-500. Notice that even though the performance is low initially at high temperature, at the later stages, they plateau around the same point. Figure 18b shows the absolute gain when decoded under different temperatures. Note that decoding with higher temperature might give the impression of a larger gain compared to low-temperature decoding. However, the evaluation curves during training resulting from SRT and RL with ground-truth look almost identical regardless of decoding temperature, which is one of our main observations in this work.

E Additional Self-Training Metrics

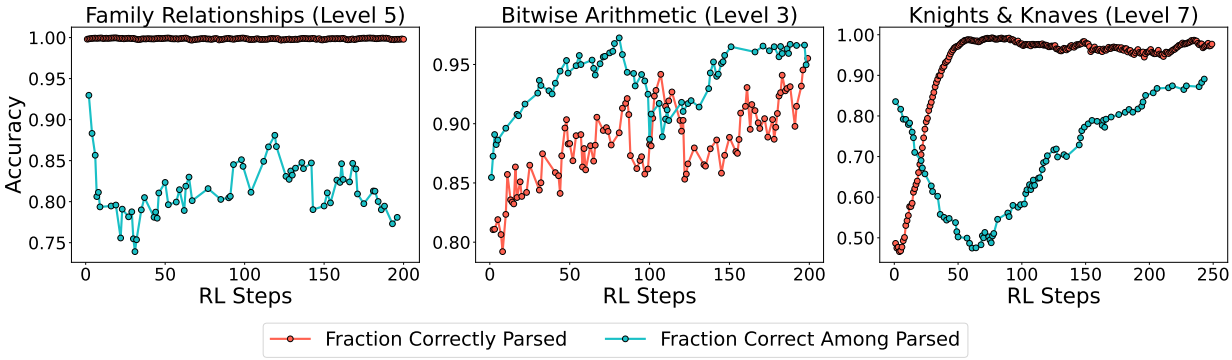


Figure 19: **(Tracking format following success rate during SRT-training on Reasoning Gym)** In order to track whether SRT is teaching reasoning strategies beyond just formatting the final answer correctly, we track two additional metrics throughout SRT-training: fraction of generations among all generations where the final answer is parseable and fraction of generations among those that are parseable where the final answer is correct. We see that due to training with RLVR on an easier level of difficulty, the starting policy can already format most generations correctly, and in the case of Knights & Knaves, fraction of correct responses keeps increasing even after fraction of properly formatted (and thus parseable) responses have saturated. This shows that the model learns reasoning strategies beyond formatting rules.

We are interested to know if SRT teaches actual reasoning strategies beyond just proper formatting rules necessary for extracting the final answer. To do so, we track two additional metrics throughout SRT-training: fraction of generations among all generations where the final answer is parseable and fraction of generations among those that are parseable where the final answer is correct. Figure 19 summarizes our findings on

Reasoning Gym: We see that due to training with RLVR on an easier level of difficulty, the starting policy can already format most generations correctly, and in the case of Knights & Knaves, fraction of correct responses keeps increasing even after fraction of properly formatted (and thus parseable) responses have saturated.

F More Details on Baselines

Baseline Implementation. For all three methods (SFT, DPO (Rafailov et al., 2024), ScPO (Prasad et al., 2024)), we sweep over three learning rates (10^{-5}, 10^{-6} and 10^{-7}) and pick the checkpoint with the highest validation score. The best checkpoint with highest validation in the SFT stage has been used to initialize the DPO/ScPO training. We also train DPO with the above mentioned learning rates and picked the best score. For DPO and ScPO (Prasad et al., 2024), we used $\beta = 0.1$, which we also found through sweep over (0.1, 0.3 and 0.5). Moreover, we add a negative log-likelihood loss with weight 1.0 to the DPO and ScPO losses to stabilize them, similar to RPO (Pang et al., 2024b). We do not train for more than 1 epoch to prevent overfitting/unintentional unalignment (Tajwar et al., 2024; Razin et al., 2025) and fair comparison with SRT.

Train Dataset	Method	AMC/AIME	MATH500
MATH	SFT	0.18	0.75
	ScPO	0.20	0.72
	DPO	0.23	0.74
	SRT (Ours)	**0.32**	**0.80**
DAPO	SFT	0.18	0.75
	ScPO	0.20	0.72
	DPO	0.21	0.76
	SRT (Ours)	**0.31**	0.75
Base Model	Accuracy	0.15	0.42
	Majority@32 Acc	0.20	0.79

Table 2: Comparison of different methods trained on either the MATH or DAPO dataset. Performance is evaluated on AMC/AIME24, 25 (average accuracy@32) and MATH500 (average accuracy@1). Notice that majority@32 accuracy scores are not directly comparable with the other accuracy metrics listed in the table.

Dataset Curation For DPO and ScPO we labeled the most consistent response as the positive example and the least consistent response as the negative example for each question. Moreover, we only kept the instances where $w(x)$, the variance based weighing parameter (Prasad et al., 2024), was greater than 2.

G Detailed Experiment Results on Different Training Settings

G.1 GRPO vs RLOO

Figure 8 shows our experiment comparing how SRT behaves with different RL algorithms. In particular, we test two algorithms: GRPO and RLOO. While GRPO seems to achieve higher performance, both GRPO and RLOO training with SRT-reward leads to model collapse at similar number of steps — showing that the choice of the RL algo does not influence model collapse.

G.2 Different KL Penalty Coefficients

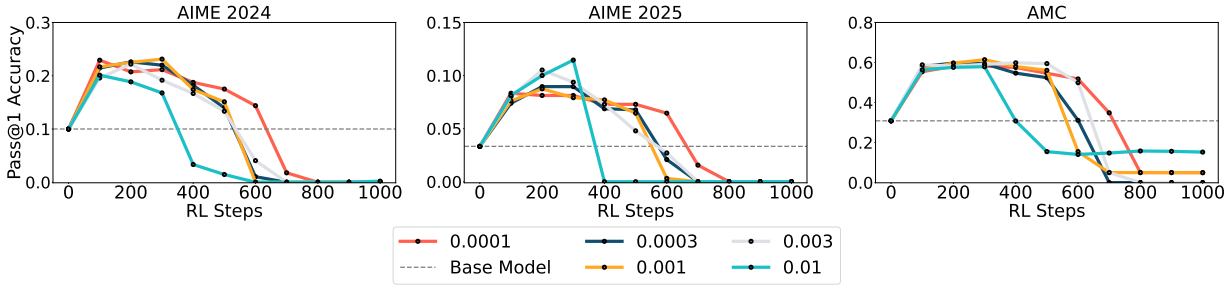


Figure 20: **(SRT with different KL penalty coefficients)** We compare the behavior of SRT with different KL penalty coefficients and report performance on all test datasets. All experiments here use a Qwen2.5-Math-7B model trained with RLOO on the DAPO dataset, with the hyperparameters other than KL penalty coefficient being the default ones described in Appendix B.3. Stronger KL penalty does not prevent or delay model collapse.

The most straightforward way of preventing reward hacking is to add a strong KL penalty to the training objective. In Figure 20, we explore this idea: to our surprise, we don't find a higher KL penalty coefficient to delay or prevent model collapse. We attribute this to the reward hacking training signal being too strong to be overcome by the KL regularization.

G.3 Learning Rate

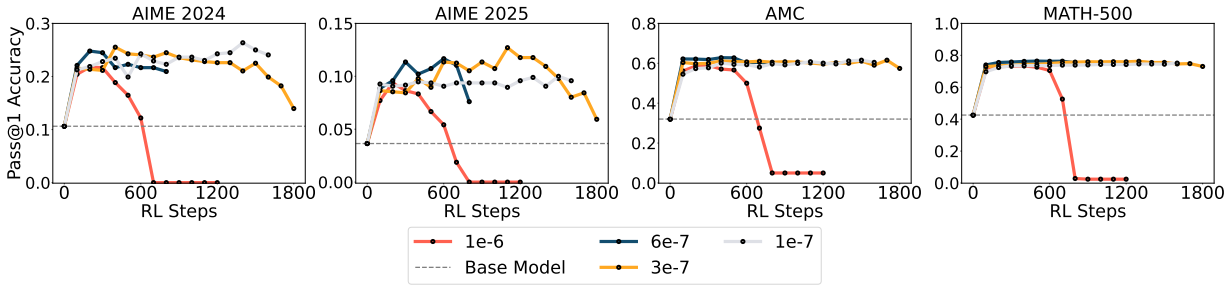


Figure 21: **(SRT with different learning rate)** We compare the behavior of SRT with different learning rate and report performance on all test datasets. All experiments here use a Qwen2.5-Math-7B model trained with RLOO on the DAPO dataset, with the hyperparameters other than learning rate being the default ones described in Appendix B.3. Lower learning rate tends to delay model collapse in our experiments — though we see performance decay on AIME 2024 and 2025 within our training budget, and hypothesize that training for longer with SRT, even with a smaller learning rate, will lead to complete model collapse as usual.

Another common hyperparameter to tune is the learning rate. To investigate the effect learning rate has on SRT, we finetune a Qwen2.5-Math-7B model with different learning rates using SRT on the DAPO dataset, with all other hyperparameters kept fixed at their default values described in Appendix B.3. Figure 21 shows our empirical findings: lowering learning rate seems to prevent model collapse within our training budget. However, we notice performance degradation on the harder AIME 2024 and 2025 datasets within our training budget, and hypothesize that prolonged training with SRT, even with a considerably lower learning rate, would still lead to model collapse. We could not study this in detail due to computational constraints, and leave this for future work.

G.4 Different Number of generations per prompt

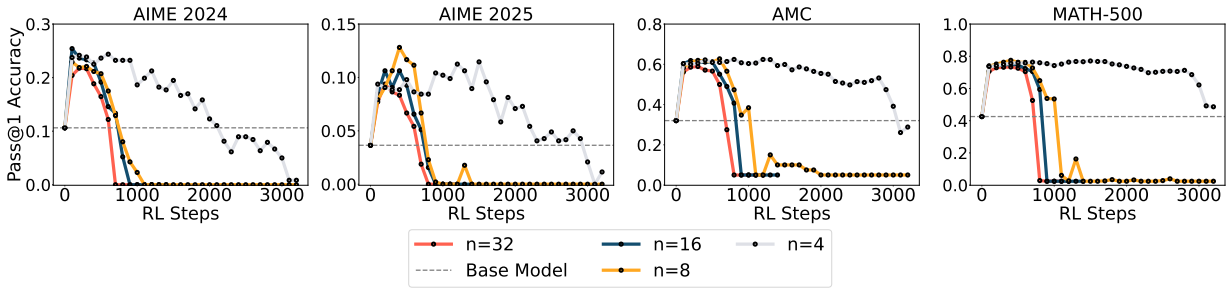


Figure 22: **(SRT with number of generations per prompt)** We compare the behavior of SRT with different number of generations/rollouts per prompt and report performance on all test datasets. All experiments here use a Qwen2.5-Math-7B model trained with RLOO on the DAPO dataset, with the hyperparameters other than number of generations per prompt kept fixed at the default ones described in Appendix B.3. Surprisingly, lowering the number of generations per prompt seems to delay model collapse.

We observe the most surprising result among our hyperparameter tuning experiments when we vary the number of rollouts per prompt during training. Similarly as before, we finetune a Qwen2.5-Math-7B model with SRT on the DAPO dataset, with all hyperparameters (except number of rollouts per prompt) kept fixed at the their default values described in Appendix B.3. Figure 22 records our results: model collapse happens progressively later in the training run as we lower the number of rollouts per prompt. We hypothesize that generating fewer rollouts makes estimating the "true" majority voting label for a single prompt more noisy. This noisy estimation then makes hacking the reward signal harder as well, thereby delaying model collapse. We have not been able to study this phenomenon in more detail due to computational constraints, and leave studying this for future work.

G.5 Entropy Coefficient

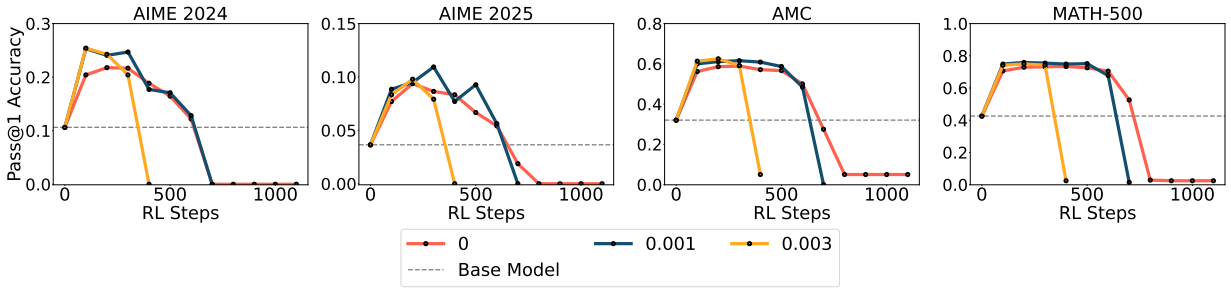


Figure 23: **(SRT with different entropy coefficient)** We compare the behavior of SRT with different entropy coefficients and report performance on all test datasets. All experiments here use a Qwen2.5-Math-7B model trained with RLOO on the DAPO dataset, with the hyperparameters other than entropy coefficient kept fixed at the default ones described in Appendix B.3. Surprisingly, increasing the entropy coefficient hastens model collapse.

The final hyperparameter with which we experiment is adding an entropy loss to our regular RL objective. The modified RL objective is listed below:

$$\mathcal{L}_{\text{entropy-augmented}}(\pi_\theta) = \mathcal{L}_{\text{RL}}(\pi_\theta) - \alpha\mathcal{H}(\pi_\theta) \tag{7}$$

where $\mathcal{L}_{\text{RL}}(\pi_\theta)$ is the regular RL loss objective, α is the entropy coefficient, and $\mathcal{H}(\pi_\theta)$ is the per-token entropy averaged across all tokens in all rollouts. A reasonable hypothesis is that adding entropy to the loss objective will prevent model collapse (Cheng et al., 2025) by discouraging the model to converge to one solution for every prompt. However, Figure 23 shows results in the contrary: adding an entropy term to the loss function and increasing the corresponding coefficient α accelerates model collapse. Upon inspection of the rollouts, we see that increasing α incentivizes the model to generate more random tokens in the rollouts, which maximizes entropy, followed by the same template final answer, which maximizes training (pseudo-)reward. This suggests that a better mechanism to prevent model diversity collapse is needed (Song et al., 2025a; Zhou et al., 2025), which we leave to future work to study.

H Test-Time Self-Improvement

H.1 SRT can be used for test-time training

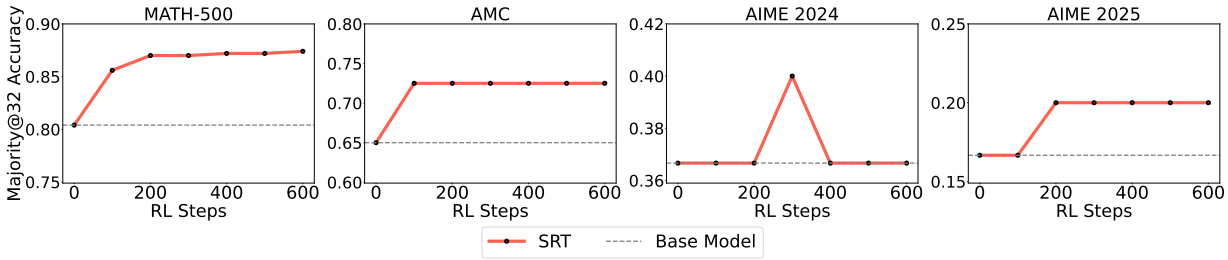


Figure 24: **(Test-Time Self-Training Performance)** Given the test dataset $\mathcal{D}_{\text{test}}$, one can perform SRT on $\mathcal{D}_{\text{test}}$ before making predictions. Our results show that this improves the majority voting performance on $\mathcal{D}_{\text{test}}$ without access to ground truth labels. Notice that the y axis is showing majority@32 accuracy instead of average@32 accuracy, for a fairer comparison with the baseline.

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 24) that test-time training via SRT provides relatively limited, yet noticeable, 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.

H.2 Why Doesn't the Performance Collapse during Test-Time-Training?

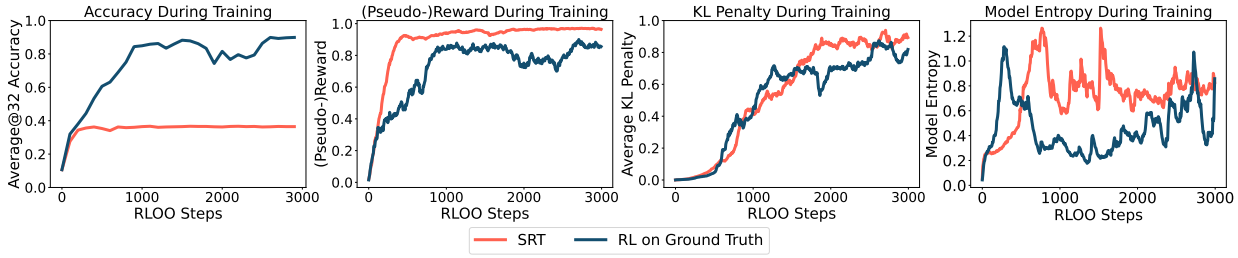


Figure 25: **(Test-Time Self-Training Dynamics)** We apply test-time training on AIME 2024 and observe no performance collapse. However, SRT's performance quickly saturates (leftmost plot), and the pseudo-reward value (second plot) also approaches saturation.

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 25 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.

I Additional Experiments with Curriculum Learning

I.1 Training Dynamics of SRT (Qwen2.5-Math-7B) on the Easy DAPO Subset

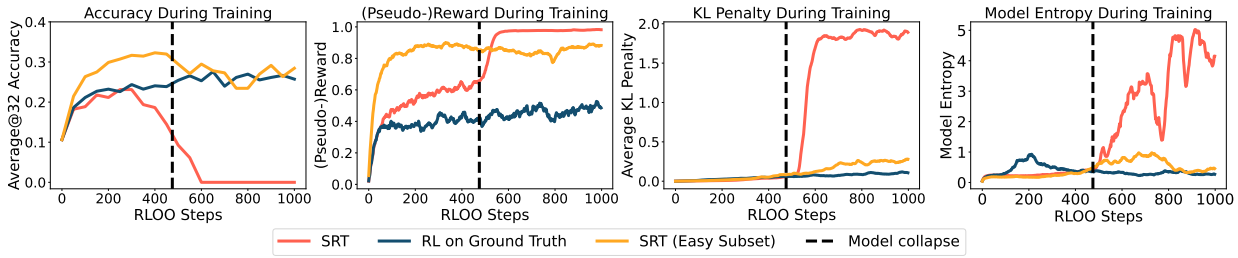


Figure 26: **(Training Dynamics of SRT on the Easy Subset of DAPO, using Qwen2.5-Math-7B)** We show the training dynamics of SRT on the easiest 1/3-rd of the DAPO dataset, chosen by ground truth pass rate of the base model. Compared to SRT on the entire DAPO dataset, SRT on the easier subset does not show any signs of reward hacking, even after taking 3 full passes over the training set.

Figure 7 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 7.

Figure 26 shows our results on Qwen2.5-Math-7B: 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 26, 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.

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

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 Figure 26 (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 27 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.

I.3 Generating Easy DAPO Subset using Qwen2.5-Math-1.5B

Next, we want to test if the easy subset generation process of our curriculum algorithm itself is reproducible using different base models. To do so, we generate the easy 1/3-rd subset using a Qwen2.5-Math-1.5B model by both majority voting frequency and pass rate. This is in contrast with our earlier sections, especially Figure 10 and Figure 27, where we used a Qwen2.5-Math-7B model for the subset generation. We also train the Qwen2.5-Math-1.5B model on the resulting easy subsets. Figure 28 shows our results — we see the

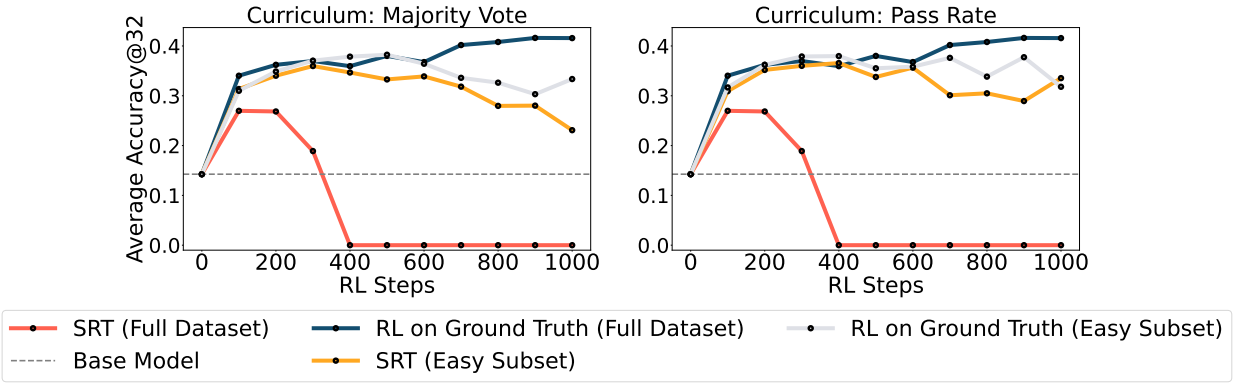


Figure 27: **(Qwen3-14B-Base Trained on the Easy DAPO Subset)** We take the same easy subsets of DAPO used in Figure 26 and train a Qwen3-14B-Base model with SRT on it. We see the same behavior as Qwen2.5-Math-7B (Figure 10), that SRT on the easy subset does not exhibit performance collapse within the same compute budget.

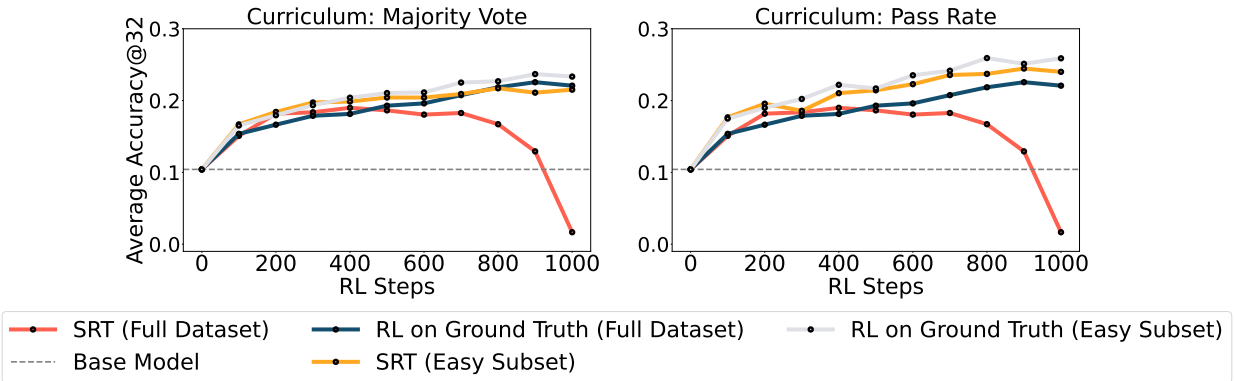


Figure 28: **(Generating Easy DAPO Subset Using Qwen2.5-Math-1.5B)** To see if the curriculum generation process is reproducible, we generate the easy 1/3-rd subset (by both majority voting frequency and pass rate) using a Qwen2.5-Math-1.5B model. This is in contrast with Figure 10 and Figure 27, where we used a Qwen2.5-Math-7B model for the easy subset generation. Furtheremore, we train the Qwen2.5-Math-1.5B on the easy subset as before, and make similar observations: training with SRT on a easy subset, even for 3 epochs, does not lead to performance collapse.

same trend as our earlier result, i.e., training on the easy subsets, even for 3 epochs, does not lead to any performance collapse. Moreover, surprisingly, up to our training budget, SRT on the easy subset matches the performance of RL training with ground truth labels.

J Detailed Experiment Results using non-Qwen models

To validate the efficacy of SRT on LLMs with different pre-training/post-training routine, we run additional experiments on two more models: Deepseek-Math-7B-Instruct (Shao et al., 2024) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Our results are described below.

J.1 Deepseek-Math-7B-Instruct

We train Deepseek-Math-7B-Instruct on the MATH-12K dataset, and test on AIME 24, AIME 2025, AMC and MATH-500. For training, we use the same hyperparameters use used to train Qwen2.5-Math-7B-Instruct due to lack of compute for running a sweep over possible hyperparameters. We note that we did not find the recommended temperature or other sampling parameters for the Instruct model in (Shao et al., 2024), but their base models were evaluated with temperature 0.7, so we choose temperature 0.7, top-p 1.0 and no top-k sampling for our evaluations. Figure 29 shows our results: we see similar trends as our earlier experiments, where SRT initially matches performance gain from RL with ground truth, but then leads to performance collapse after prolonged training.

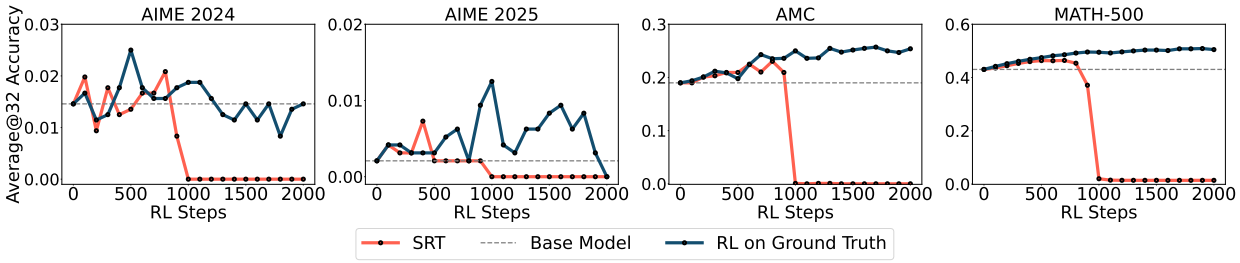


Figure 29: **(Training Deepseek-Math-7B-Instruct on MATH-12K using SRT)** We see similar trends for SRT-training on Deepseek-Math-7B-Instruct as we saw on our experiments with Qwen models: SRT initially matches performance gain obtained with RL training with ground truth, but faces performance collapse after prolonged training.

J.2 Llama-3.1-8B-Instruct

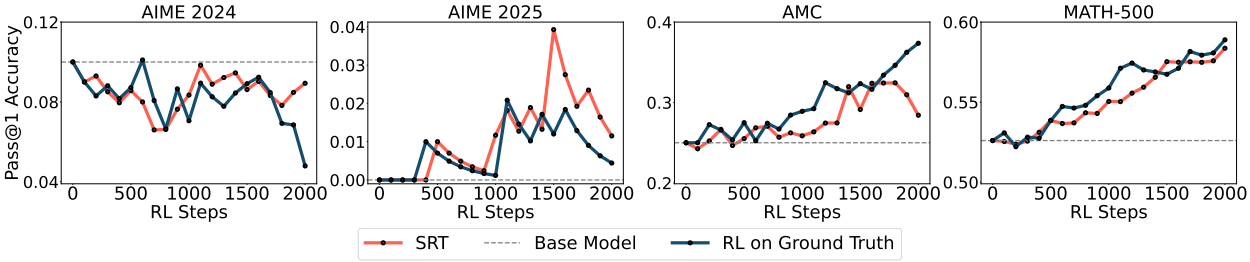


Figure 30: **(Training Llama-3.1-8B-Instruct on Big-Math-RL-Verified with learning rate 10^{-7})** Llama-3.1-8B-Instruct, when trained on a filter subset of the Big-Math dataset, shows significant gains on MATH-500 from both SRT and RL with ground truth. In fact, up to our training budget of 2K steps, both seem to improve performance at the same rate, from 52.6% to around 60%.

Llama-3.1-8B-Instruct showed no gains while being trained on DAPO or MATH-12K. This can be due to insufficient hyperparameter tuning or the model's starting performance on these datasets not suitable for learning. So we chose the Big-Math dataset (Albalak et al., 2025), a dataset with over 250,000 math questions with verifiable answers. Moreover, this dataset has been constructed by filtering common evaluation datasets like MATH-500, making it suitable for our purposes. **The primary benefit of using this dataset is that it comes with Llama-3.1-8B-Instruct pass rate**, so we can easily ascertain the difficulty of each datapoint and aggregate a subset that can be suitable for training the Llama model. Specifically, we take the subset of Big-Math where Llama-3.1-8B-Instruct has pass rate between 0.3 and 0.7, to filter away too easy or too difficult questions. Next, we train the model on this subset using the same hyperparameters as

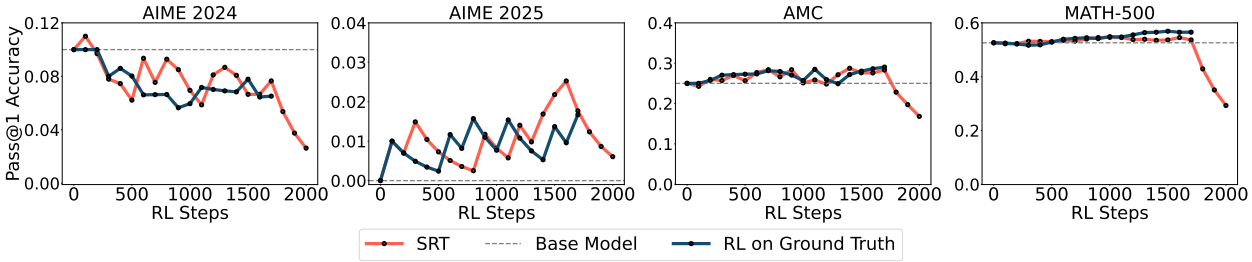


Figure 31: **(Training Llama-3.1-8B-Instruct on Big-Math-RL-Verified with learning rate 3×10^{-7})** Llama-3.1-8B-Instruct, when trained on a filter subset of the Big-Math dataset with a higher learning rate (3×10^{-7} instead of 10^{-7} in Figure 30) demonstrates model collapse within the same training budget.

we used for training the Qwen2.5-Math-7B-Instruct, except we lower the learning rate to 10^{-7} (from 10^{-6}), as we see that leads to more stable learning curves. For evaluation, we use the same prompt template but temperature 0 (greedy decoding), to match the starting model's performance reported on its model card, and report pass@1 accuracy.

Figure 30 shows our results: on MATH-500, Llama-3.1-8B-Instruct shows the same performance growth when trained via SRT or RL with ground truth, up to our training budget of 2000 steps. Performance growth is also significant, and training improves pass@1 accuracy from 52.6% to around 60%. Note that we do note report performance on the harder datasets like AIME, because the Llama model's performance remain close to 0 throughtout training (with both objectives) on these datasets, signalling that they might be too hard for this model. We also ran an additional experiment using a higher learning rate of 3×10^{-7}. Figure 31 shows our empirical findings: with the higher learning rate, Llama-3.1-8B-Instruct also start to show model collapse within our training budget.

K Example Tasks from Reasoning Gym

Task: Family Relationships (Level 4)

Question: John is married to Isabella. They have a child called Edward. Edward is married to Victoria. What is Isabella to Edward? Respond only with the word that describes their relationship.

Answer: mother

Task: Bitwise Arithmetic (Level 2)

Question: Please solve this problem. Assume there is arbitrary bit depth and that there are signed integers. If the answer is negative, reply as a negative value (e.g., $-0x3$), not the two's-complement form. Reply only with the final hexadecimal value.

$$((0x3a24 - 0x24b8) + (0x1741 \gg 0x3))$$

Answer: 0x1854

Task: Knights and Knaves (Level 2)

Question: A very special island is inhabited only by sages and fools. Sages always tell the truth, and fools always lie. You meet 2 inhabitants: Zoey and Riley. Zoey commented, "Riley is a fool." In Riley's words: "Zoey is a sage or Riley is a sage." So who is a sage and who is a fool? (Format your answer like: "Zoey is a sage/fool, and Riley is a sage/fool")

Answer: Zoey is a fool, and Riley is a sage.

Above of we see one example from each of the three Reasoning Gym tasks used in our work. The examples shown are of the lowest difficulties that we first train the model on using ground truth RL. We do self training on more difficult variants on the tasks. Difficulty can be changed by the modifying the either number of person or digits. In this work, we abstract it away by calling it "level".