TTRL: Test-Time Reinforcement Learning

Yuxin Zuo[*1,2] **Kaiyan Zhang**[*1] **Li Sheng**[1,2] **Shang Qu**[1,2] **Ganqu Cui**[2]
Xuekai Zhu[1] **Haozhan Li**[1,2] **Yuchen Zhang**[2] **Xinwei Long**[1] **Ermo Hua**[1]
Biqing Qi[2] **Youbang Sun**[1] **Zhiyuan Ma**[1] **Lifan Yuan**[1]
Ning Ding[†1,2] **Bowen Zhou**[†1,2]
[1]Tsinghua University [2]Shanghai AI Lab
zhang-ky22@mails.tsinghua.edu.cn, dingning@mail.tsinghua.edu.cn

Code: https://github.com/PRIME-RL/TTRL

Abstract

This paper investigates Reinforcement Learning (RL) on data *without* explicit labels for reasoning tasks in Large Language Models (LLMs). The core challenge of the problem is reward estimation during inference while not having access to ground-truth information. While this setting appears elusive, we find that common practices in Test-Time Scaling (TTS), such as majority voting, yield surprisingly effective rewards suitable for driving RL training. In this work, we introduce Test-Time Reinforcement Learning (**TTRL**), a novel method for training LLMs using RL on unlabeled data. **TTRL** enables self-evolution of LLMs by utilizing the priors in the pre-trained models. Our experiments demonstrate that **TTRL** consistently improves performance across a variety of tasks and models. Notably, **TTRL** boosts the pass@1 performance of `Qwen-2.5-Math-7B` by approximately **211**% on the AIME 2024 with only unlabeled test data. Furthermore, although **TTRL** is only supervised by the maj@n metric, **TTRL** has demonstrated performance to consistently surpass the upper limit of the initial model maj@n, and approach the performance of models trained directly on test data with ground-truth labels. Our experimental findings validate the general effectiveness of **TTRL** across various tasks and highlight **TTRL**'s potential for broader tasks and domains.

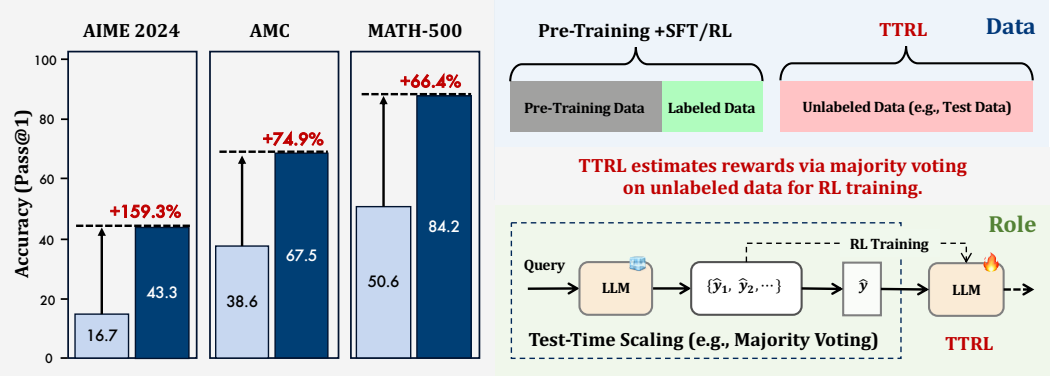


Figure 1: Performance and Position of **TTRL**.

[*]Equal Contributions. [†]Corresponding Authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

1 Introduction

Recent advances in Large Reasoning Models (LRMs), such as DeepSeek-R1 [11] and OpenAI's o1 [18], have demonstrated that Reinforcement Learning (RL) is essential for enhancing long chain-of-thought (CoT) reasoning [49] through training on expensive human-annotated data. These models achieve remarkable performance on a range of highly challenging tasks. For example, OpenAI's o3 attains a 75.7% success rate on ARC-AGI-1. However, complex and unlabeled questions continuously emerge, posing significant challenges. For instance, o3 solves only 4% of problems on the recently released ARC-AGI-2 benchmark (2025). Addressing such tasks typically involves scaling up training with more data and computational resources, and it may still fail to yield strong performance on these tasks. Silver & Sutton [39] has recently advocated for a transition to the "era of experience," emphasizing the limitations of existing AI systems that rely heavily on human supervision, as well as the importance of enabling models to self-evolve through experience.

Further building upon the substantial progress of LRMs, it naturally motivates a promising direction in which AI systems autonomously improve via RL on unlabeled data by directly engaging in self-experience and learning, thereby pushing the boundaries of RL and further advancing the frontier of AI capabilities. Such self-evolvement can be broadly categorized into two modes: adaptation to test-time data, which enables models to tackle harder benchmarks such as ARC-AGI-2, and training on external unlabeled data, which unlocks more training data beyond labeled corpora. This work focuses on the adaptation to test-time data, which has been extensively studied under the paradigm of Test-Time Training (TTT) [1, 2, 42, 43]. TTT has received increasing attention recently. These approaches adapt model parameters at test time by exploiting the structure and distributional properties of incoming test data.

Therefore, we aim to fully advance AI evolution by updating models at test time using RL, thereby enhancing their generalization to previously unseen data. However, this introduces a critical challenge: *How to obtain rewards for RL at test-time?* This also highlights a broader limitation of current RL approaches. Despite their promise, most existing methods still rely heavily on labeled data, which significantly limits their scalability. As real-world tasks continue to increase in both complexity and volume, large-scale annotation for RL becomes increasingly impractical, posing a substantial barrier to the continual improvement of state-of-the-art models.

We introduce Test-Time Reinforcement Learning (**TTRL**), which performs test-time training through RL. **TTRL** employs repeated sampling strategies in the rollout phase to accurately estimate the label and compute rule-based rewards, thereby enabling RL on unlabeled data. By incorporating effective majority voting rewards, **TTRL** facilitates efficient and stable RL in the absence of ground truth labels. As previously highlighted, the emergence of more challenging tasks will inevitably lead to larger proportions of unlabeled data. **TTRL** directly addresses the problem of training models via RL without explicit supervision, investigating a model's ability to explore and learn in this challenging yet critical setting. Essentially, **TTRL** enables the model to generate its own experiences, estimate rewards, and improve its performance over time.

In experiments, applying **TTRL** to `Qwen2.5-Math-7B` results in an improvement on AIME 2024 of $\mathbf{211}\%$ (12.9 to 40.2), with an average gain of $\mathbf{76}\%$ across AIME 2024, AMC, MATH-500, and GPQA. These improvements are achieved through self-evolution without any labeled training data and further generalize to other tasks. **TTRL** not only enhances performance on pass@1 but also improves TTS through majority voting. Moreover, our preliminary experiments suggest that **TTRL** is effective across models of different scales and types and that it can be integrated with existing RL algorithms. We also found that **TTRL** exhibits favorable characteristics such as a high-performance ceiling. These observations highlight its potential to substantially reduce reliance on human annotations, enabling continual learning and scaling RL to large-scale unsupervised training. Below are several key takeaways:

Takeaways

1. Majority voting provides effective reward estimation for **TTRL** (**§ 3**).
2. **TTRL** can exceed its training signal and upper limit maj@n, and closely mirrors the performance of direct training on the test data with ground-truth (**§ 4.1**).
3. It is possible to achieve efficient and stable RL in an unsupervised manner (**§ 4.2**).

2 Test-Time Reinforcement Learning (TTRL)

Unlike traditional RL, where the agent learns from known reward signals, **TTRL** operates on unlabeled test data. In other words, the model must learn and adapt without access to explicit supervision. Our task is defined as follows:

> We study the problem of training a pre-trained model during test time using RL without ground-truth labels. We call this setting Test-Time Reinforcement Learning.

2.1 Methodology

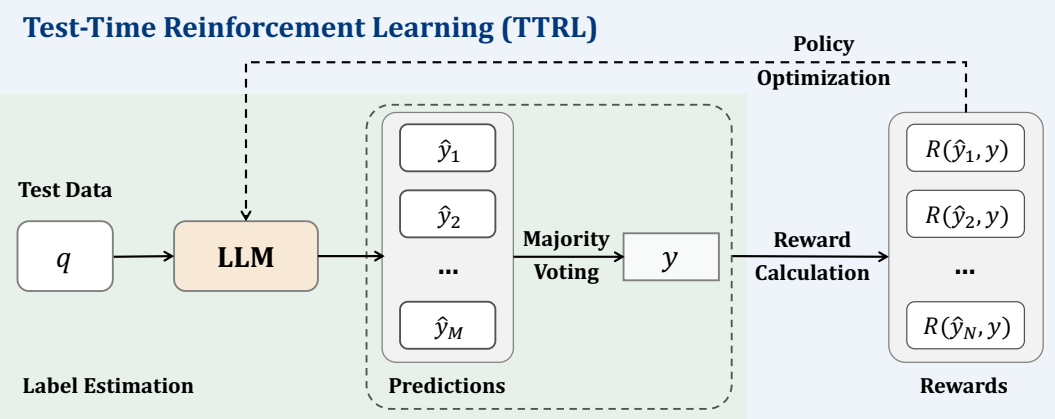


Figure 2: **TTRL** combines both Test-Time Scaling (TTS) and Test-Time Training (TTT).

Figure 2 illustrates how our approach, **TTRL**, tackles this challenge. Given a state represented by the prompt x, the model acts by producing an output y sampled from a policy $\pi_\theta(y \mid x)$ parameterized by θ. To construct a reward signal without ground-truth labels, we generate multiple candidate outputs $\{y_1, y_2, \ldots, y_N\}$ from the model through repeated sampling. A consensus output y^* is derived, for instance, by majority voting or another aggregation method, serving as a proxy for the optimal action. The environment then provides a reward $r(y, y^*)$ based on the alignment between the sampled action y and the consensus action y^*. The RL objective is thus to maximize the expected reward:

$$\max_\theta \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[r(y, y^*)], \tag{1}$$

and parameters θ are updated through gradient ascent:

$$\theta \leftarrow \theta + \eta \nabla_\theta \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[r(y, y^*)], \tag{2}$$

where η denotes the learning rate. This approach enables the model to adapt during inference, effectively improving its performance on distribution-shifted inputs without the need for labeled data.

2.2 Majority Voting Reward Function

The majority voting reward is determined by first estimating a label through majority voting. This estimated label is then used to calculate rule-based rewards, which serve as the final rewards. Given a question x, we first input x into the LLM to generate a set of outputs. An answer extractor then processes these outputs to obtain the corresponding predicted answers, denoted as $P = \{\hat{y}_i\}_{i=1}^N$. We first follow Equation 4 over P to estimate a label, with majority voting as the scoring function $s(y, x)$ to get y, the most frequently occurring prediction in P. The majority-voted prediction y is then used as the estimated label to compute rule-based rewards [11]. The reward function is:

$$R(\hat{y}_i, y) = \begin{cases} 1, & \text{if } \hat{y}_i = y, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Appendix C presents the pseudo-code of the reward function.

Table 1: Main results of **TTRL** on each task. * indicates that Qwen3-8B is evaluated in non-thinking mode within a 3k context. Figure 3 provides results within a 32k context.

Name	AIME 2024	AMC	MATH-500	GPQA	Avg
Math Base Models					
`Qwen2.5-Math-1.5B`	7.7	28.6	32.7	24.9	23.5
w/ **TTRL**	15.8	48.9	73.0	26.1	41.0
Δ	+8.1	+20.3	+40.3	+1.2	+17.5
	↑ 105.2%	↑ 71.0%	↑ 123.2%	↑ 4.8%	↑ 74.4%
`Qwen2.5-Math-7B`	12.9	35.6	46.7	29.1	31.1
w/ **TTRL**	40.2	68.1	83.4	27.7	54.9
Δ	+27.3	+32.5	+36.7	−1.4	+23.8
	↑ 211.6%	↑ 91.3%	↑ 78.6%	↓ 4.8%	↑ 76.5%
Vanilla Base Models					
`Qwen2.5-7B`	7.9	34.8	60.5	31.8	33.8
w/ **TTRL**	23.3	56.6	80.5	33.6	48.5
Δ	+15.4	+21.8	+20.0	+1.8	+14.7
	↑ 194.9%	↑ 62.6%	↑ 33.1%	↑ 5.7%	↑ 43.7%
`Qwen2.5-32B`	7.9	32.6	55.8	33.2	32.4
w/ **TTRL**	24.0	59.3	83.2	37.7	51.1
Δ	+16.1	+26.7	+27.4	+4.5	+18.7
	↑ 203.8%	↑ 81.9%	↑ 49.1%	↑ 13.6%	↑ 57.7%
Instruct Models					
`LLaMA3.1-8B`	4.6	23.3	48.6	30.8	26.8
w/ **TTRL**	10.0	32.3	63.7	34.1	35.0
Δ	+5.4	+9.0	+15.1	+3.3	+8.2
	↑ 117.4%	↑ 38.6%	↑ 31.1%	↑ 10.7%	↑ 30.6%
`Qwen3-8B`*	26.9	57.8	82.3	48.1	53.8
w/ **TTRL**	46.7	69.1	89.3	53.0	64.5
Δ	+19.8	+11.3	+7.0	+4.9	+10.8
	↑ 73.6%	↑ 19.6%	↑ 8.5%	↑ 10.2%	↑ 20.0%

3 Experiments

3.1 Experimental Setup

Models To evaluate the generalizability of **TTRL** across different backbone models, we conduct experiments using both base and instruct models of various scales. In addition, we carry out experiments on leading LRMs to demonstrate that **TTRL** can improve model performance even after costly post-training. The models we experiment with are as follows:

- **Qwen Family:** Qwen2.5-Math-1.5B [54], Qwen2.5-Math-7B [54], Qwen2.5-7B [55], Qwen2.5-32B [55], Qwen3-8B (thinking mode & non-thinking mode) [55];
- **LLaMA Family:** LLaMA-3.1-8B-Instruct [10], LLaMA-3.2-3B-Instruct [10], LLaMA-3.2-3B-Oat-Zero [26];
- **Mistral Family:** Mistral-Nemo-Instruct-2407 [29], Ministral-8B-Instruct-2410 [28];
- **DeepSeek Family:** DeepSeek-Math-7B-Instruct [38], DeepSeek-R1-LLaMA-8B [11];
- **Others:** Skywork-OR1-Math-7B [13];

Benchmarks We evaluate **TTRL** on GPQA-Diamond [35], a challenging and high-quality subset of the Graduate-Level Google-Proof Question Answering benchmark, and 3 mathematical reasoning benchmarks: AIME 2024 [21], AMC [21], and MATH-500 [14].

Evaluation Setup We apply **TTRL** to each benchmark individually and then evaluate. We set the maximum generation length to 3072 tokens, unless otherwise specified. For the **main experiments**, following DeepSeek-R1 [11], we adopt the pass@k evaluation protocol [3] and calculate pass@1 using non-zero temperature sampling. Specifically, we generate 16 responses (4 for 32k context) per question using a temperature of 0.6 and a top-p value of 0.95. For the **analysis and additional experiments on** `Qwen2.5-MATH`, we evaluate using greedy decoding to report pass@1, to ensure a

fair comparison with previous works. Appendix E presents a set of training-time metrics we used to monitor the performance of **TTRL** and analyze its training dynamics in the absence of ground-truth.

Baselines Since the use of TTT for reasoning has not been previously explored, we primarily compare it with the backbone model to validate whether **TTRL** can achieve effective improvements through self-evolution. Appendix D presents additional experimental results comparing **TTRL** with previous state-of-the-art RL approaches for reasoning.

Implementation Details We independently apply GRPO [38] on each benchmark to implement **TTRL**. For hyperparameters, we use a cosine learning rate schedule with a peak value of 5×10^{-7} and adopt the AdamW optimizer for the policy model. For rollout, we sample 64 responses using a temperature of 0.6 (1.0 for `Qwen2.5-Math` and LRMs) for voting-based label estimation and downsample 32 responses per prompt for training. Evidence shows that our vote-then-sample strategy effectively reduces computational costs while still achieving strong performance. The maximum generation length is set to 32,768 tokens for LRMs and 3,072 tokens for all other models. We set the number of episodes to 10, 30, and 80 for MATH-500, AMC, and AIME 2024, respectively, based on the dataset size. All experiments were conducted on 8 * NVIDIA A100 80GB GPUs.

3.2 Main Results

TTRL performs well on most tasks and models. Table 1 presents the main results. We apply **TTRL** to 6 models spanning 4 model families, 2 model types, and 3 model sizes, consistently demonstrating substantial improvements across 4 highly challenging benchmarks. On the demanding mathematical reasoning benchmark AIME 2024, **TTRL** achieves a minimum improvement of 105% across all 6 models. Moreover, applying **TTRL** to a 1.5B model leads to a significant gain of up to 40.3 points on the MATH-500. Recently, Shao et al. [37] demonstrated the importance of evaluating different models for RL-based methods to validate experimental conclusions. Therefore, we additionally report results on a broader range of models from various model families, such as DeepSeek-R1-LLaMA-8B, an LRM from DeepSeek trained on the LLaMA model. Table 2 presents the results. As shown, **TTRL** continues to exhibit consistent effectiveness. Furthermore, as shown in Appendix D, despite relying solely on self-evolution using unlabeled test data, **TTRL** achieves performance comparable to existing RL-based models that are trained on large-scale labeled datasets.

Table 2: Performance of **TTRL** on various models.

Name	AIME	AMC	MATH-500
LLaMA Family			
LLaMA-3.2-3B-Oat-Zero	0.8	15.1	41.9
w/ **TTRL**	3.3	25.3	55.7
Δ	+2.5	+10.2	+13.8
LLaMA-3.2-3B-Instruct	6.0	19.4	43.9
w/ **TTRL**	13.3	31.3	61.6
Δ	+7.3	+11.9	+17.7
Mistral Family			
Mistral-Nemo-Instruct	0.8	15.4	40.8
w/ **TTRL**	0	24.8	51.0
Δ	−0.8	+9.4	+10.2
Ministral-8B-Instruct	1.3	19.7	52.4
w/ **TTRL**	3.3	28.9	57.8
Δ	+2.0	+9.2	+5.4
DeepSeek Family			
DeepSeek-Math-7B-Instruct	1.9	16.3	42.3
w/ **TTRL**	2.5	22.9	52.4
Δ	+0.6	+6.6	+10.1
DeepSeek-R1-LLaMA-8B	51.7	81.6	89.6
w/ **TTRL**	69.2	88.9	90.9
Δ	+17.5	+7.3	+1.3

TTRL performs well on LRMs. With the rapid progress in RL and TTS, LRMs are becoming increasingly central. To further examine whether **TTRL** remains effective on LRMs that have undergone expensive post-training, especially on highly challenging tasks, we evaluate two other powerful LRMs. Figure 3 presents the results of applying **TTRL** to additional reasoning models. `Qwen3-8B` is evaluated in thinking mode. Despite the extensive post-training these models have undergone, **TTRL** still achieves substantial performance gains, yielding improvements of approximately 10 points on both backbones.

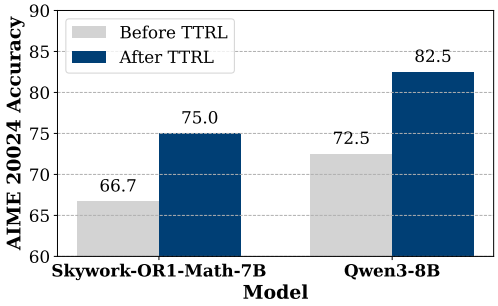


Figure 3: **TTRL** on LRMs.

TTRL naturally scales. Another noteworthy observation is that as the model size increases (1.5B $\rightarrow$ 7B and 7B $\rightarrow$ 32B), performance consistently improves, highlighting the natural scaling behavior of **TTRL**: larger models can produce more accurate majority voting rewards during self-improvement, which leads to more effective learning on new data.

TTRL generalizes well beyond the target task. We perform **TTRL** on each benchmark and further evaluate pass@1 using greedy decoding on others, with `Qwen2.5-Math-7B` as the backbone. Figure 4 shows the results. Despite the out-of-distribution nature of this setting, **TTRL** achieves substantial improvements across all benchmarks. This suggests that **TTRL** does not rely on overfitting, which would lead to trade-offs on other tasks, but instead acquires generalizable gains during self-improvement.

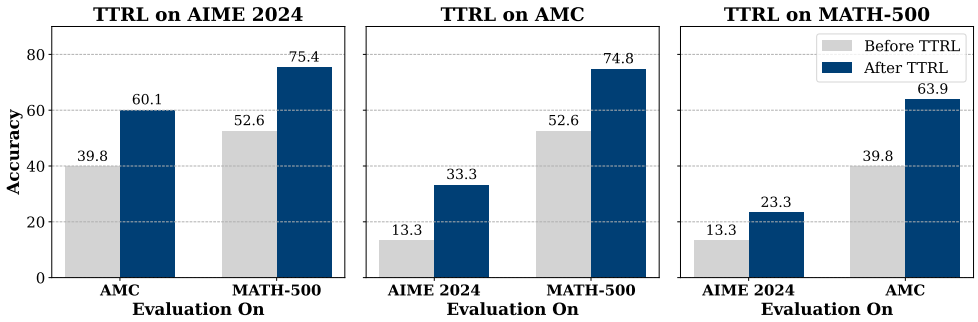
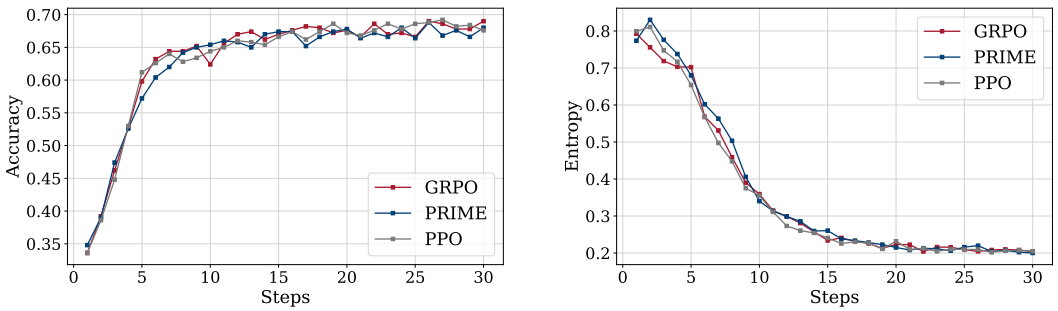


Figure 4: Out-of-distribution performance before and after **TTRL**.

TTRL is compatible with different RL algorithms. We further apply **TTRL** using two RL algorithms on MATH-500 with `Qwen2.5-Math-1.5B` to assess its compatibility, which are PPO [36], a value mode based method, and PRIME [7], a process-level RL algorithm. Figure 5 presents the results. The performance trajectories of GRPO, PPO, and PRIME are closely aligned.



(a) Accuracy Curve.

(b) Entropy Curve.

Figure 5: Comparison over steps of different RL algorithms, GRPO, PPO, and PRIME on MATH-500.

TTRL achieves sustainable self-evolution through "online" and "RL". To gain a deeper understanding of the underlying mechanisms of **TTRL**, we conduct an analysis of the model's training dynamics by tracking the average (pass@1/avg@16) and majority (maj@16) scores throughout the training process. Given that majority voting serves as the basis for generating training signals, examining its performance trajectory is essential for understanding how it functions. Furthermore, we investigate whether **TTRL** improves pass@1 at the cost of a reduction in maj@16 performance. Figure 6 illustrates the **TTRL** training dynamics on AMC with `Qwen2.5-Math-1.5B` as the base model. **It is notable that, as training progresses, both metrics demonstrate a consistent upward trend.** This indicates that **TTRL** is not simply approaching the initial model's majority voting performance. Due to its dynamic nature, **TTRL** can generate higher-quality supervision signals as its capabilities improve. Moreover, through **TTRL**'s use of RL for TTT, by converting voting-based pseudo-labels into reward signals, it enhances the effective supervision quality (e.g., accuracy; see Q2 4.2), while decoupling learning from the limitations imposed by maj@n.

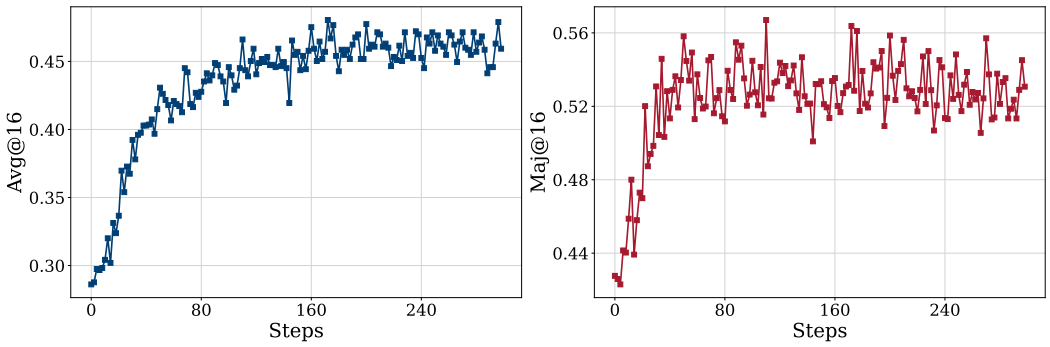


Figure 6: Training dynamics of **TTRL** on AMC using `Qwen2.5-Math-1.5B` as the base model.

4 Analysis and Discussions

4.1 Q1: How Well Can TTRL Perform?

Takeaways

1. **TTRL** surpasses the traditional self-training upper bound, the majority accuracy of the initial model.
2. The empirical upper bound of **TTRL** is direct RL on labeled test data (i.e., training on the test data). **TTRL** can approach the performance of this upper bound, highlighting its potential advantages in efficacy over standard *training-evaluation* protocols.
3. For challenging tasks, **TTRL** can reach the empirical upper bound using only a 1.5B model. This demonstrates that LLMs can now efficiently self-evolve through **TTRL**, enabling unbounded lifelong learning on large-scale datasets.

We analyze the potential performance of **TTRL** using two upper bounds. The first upper bound is the maj@n of the initial model. The second upper bound is direct training on benchmark data, which assumes access to ground-truth labels and thus leaks label information to the policy model.

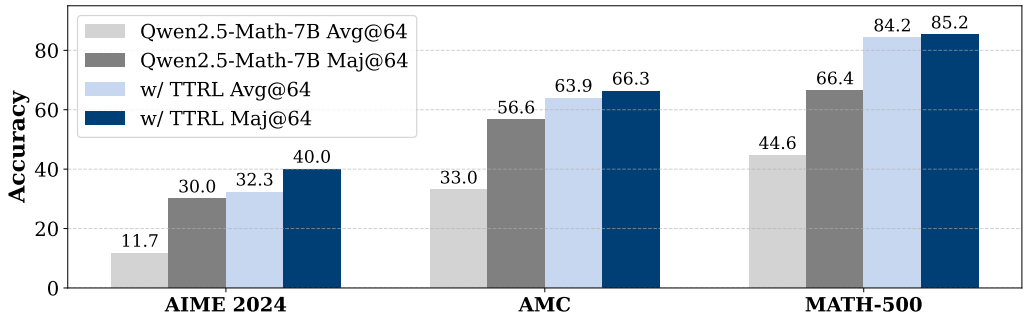


Figure 7: Majority voting performance comparison between the backbone and after **TTRL**.

TTRL is Supervised by maj@n Yet Surpasses It. Since **TTRL** utilizes the model's own majority-voted outputs for RL, this voting-based performance of the initial model can intuitively be regarded as an upper bound of the final performance. This upper bound is also the performance limit of traditional self-training methods [17], which select self-generated CoT through majority voting for supervised fine-tuning (SFT). However, we observe a surprising phenomenon: after training, the model not only matches but also surpasses the expected upper bound, suggesting that it exceeds the performance limit of the original model, which also serves as its initial supervision signal. Figure 6 illustrates this remarkable result, where it can be observed that the final avg@16 score exceeds the initial maj@16 score by more than 20 points. Furthermore, we perform additional evaluations of **TTRL** on `Qwen2.5-Math-7B` across various benchmarks, using more samples per question to enable more

reliable assessment. Figure 7 shows results. **It can be observed that TTRL avg@64 consistently outperforms `Qwen2.5-Math-7B` maj@64 across all benchmarks, with a considerable margin.** Through a self-reinforcing loop, the model ***"lifts itself up by its own bootstraps"***, evolving beyond the performance ceiling. Moreover, the performance of **TTRL** further improves with majority voting.

TTRL's Performance Gains Approach Training on the Benchmark. The motivation of **TTRL** is to estimate labels using majority voting to obtain more accurate rewards, facilitating effective self-improvement through RL on the data without ground-truth labels. Therefore, a natural upper bound of **TTRL** is performing RL directly on the test data, denoted as RL (leakage). Although this setting is rarely adopted or studied due to the issue of information leakage, it represents the most efficient way to improve performance on the particular dataset, with efficiency that far exceeds traditional training-evaluation paradigms. We use `Qwen2.5-Math-7B` to perform both **TTRL** and RL (leakage) on MATH-500 and conduct evaluations. Figure 8 shows results. Surprisingly, we find that the performance curve of **TTRL** closely approaches that of RL (leakage). This suggests that:

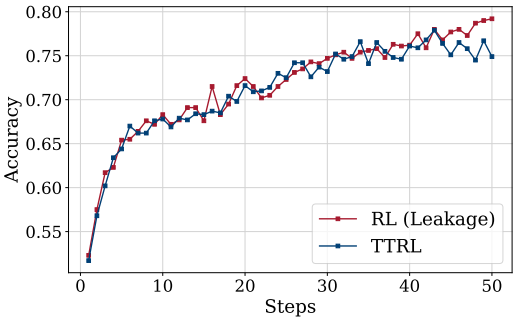


Figure 8: Comparison of RL (Leakage) *vs* **TTRL**.

1. **TTRL** can achieve a level of self-improvement comparable to that of supervised learning (even in the information leakage scenario) through RL in an unsupervised setting.

2. **TTRL** provides evidence that even small LLMs can now effectively self-improve on input-only challenging tasks through RL, enabling continual learning. Results on `Qwen2.5-Math-1.5B` further support this observation: starting from a subpar performance of 32.7 on MATH-500, the model improved by $\mathbf{123.2}\%$ to reach 73.0, demonstrating clear self-improvement through **TTRL**.

4.2 Q2: Why Does TTRL Work?

This section presents an analysis of the factors enabling **TTRL** to achieve stable and effective RL under unsupervised conditions. Our analysis identifies three key factors: reward calculation, label estimation, and online learning.

Reward Calculations. When the model is capable of estimating accurate labels via majority voting, the reward and subsequently training are generally reliable. However, a natural question arises: **Why does TTRL remain effective even when the model fails to estimate accurate labels via majority voting on challenging benchmarks such as AIME 2024?** The most fundamental reason lies in the mechanism by which the verifier computes rewards in RL. For tasks such as mathematics, the verifier works based on "comparison" to obtain rule-based rewards by checking whether the predicted answer matches the given "label." This mechanism can lead to the phenomenon of "Lucky Hit": for an incorrectly predicted answer, even if the estimated label does not match the ground truth label, as long as it differs from the predicted answer, the verifier will still output a negative reward, and this is exactly the correct reward that we expect, as illustrated in Figure 9. In other words, it is sufficient that the estimated label differs from the predicted answer for the verifier to assign the correct negative reward. To provide a more detailed case study, we examine the performance of **TTRL** on the AIME 2024 using `Qwen2.5-Math-7B`. Figure 10 presents the variation curves of the three metrics, as described in Appendix E. **We identify two main reasons why TTRL remains effective on AIME 2024:**

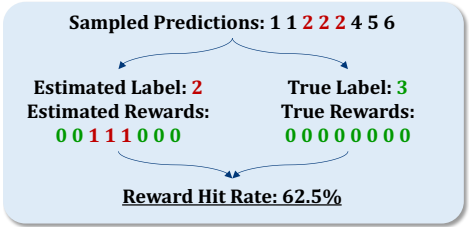


Figure 9: A toy case of "Lucky Hit". We illustrate a basic numerical prediction scenario to compare reward computation under two conditions: when the model incorrectly estimates the label versus when the ground-truth label is used. As shown on the left, although the estimated label is incorrect, some of the incorrect predictions still differ from the wrong label and therefore receive the correct reward (denoted as 0).

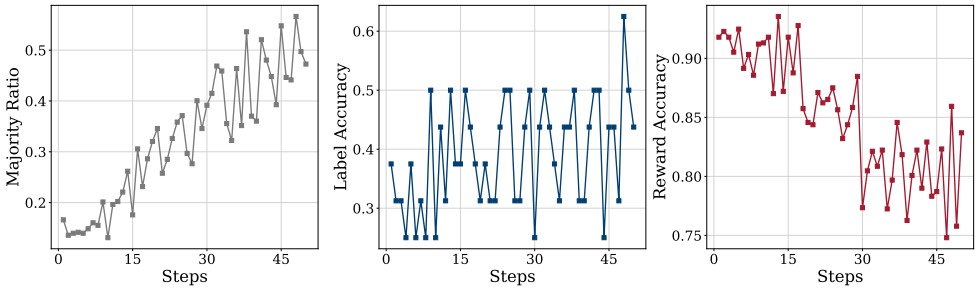


Figure 10: Comparison of Majority Ratio, Label Accuracy, and Reward Accuracy on AIME 2024 over steps. Even with low label accuracy, reward accuracy remains high due to "Lucky Hit", allowing **TTRL** to provide reliable training signals.

1. **Reward robustness enabled by multiple outputs within a rollout.** First, rewards are denser than labels, allowing for more opportunities to recover useful reward signals even when the estimated label is inaccurate. For example, even when the predicted label is incorrect, alternative outputs within the same rollout can still yield correct or high-quality rewards, as shown in Figure 9, whereas a rollout containing only a single output would not provide such flexibility. This makes the overall reward signal more robust to errors in pseudo-label estimation.
2. **High reward accuracy due to scattered incorrect predictions.** Second, counterintuitively, when the model has weaker capability, the majority voting rewards of **TTRL** may be more accurate. As shown in Figure 10, although the initial label estimation through majority voting achieves an accuracy of only 37%, the reward accuracy reaches an impressive 92%. By examining the model outputs, we find that this is because the model's responses are highly scattered and consistently incorrect, as shown in Figure 9. A result consistent with this observation is that, for the base model, the most frequently predicted answer accounts for only 16.6% of all predictions, indicating that the outputs are highly scattered. Therefore, even when the labels are not accurately estimated, due to "Lucky Hit", most outputs can still receive correct rewards. Moreover, the poorer the model's performance, the more mistakes it tends to make, which paradoxically leads to more accurate reward estimation. An empirical observation supporting this view is the comparison between the label accuracy and reward accuracy, as shown in Figure 10. Although the label accuracy rarely exceeds 50%, the reward accuracy remains consistently high, staying above 75%. This high reward accuracy provides a reliable foundation for effective self-improvement on test data.

Label Estimations. A direct difference between **TTRL** and standard RL algorithms is that **TTRL** involves label estimation, which introduces reward inaccuracies. We believe that **TTRL** works despite these inaccuracies due to the following two reasons. **(i)** Existing studies have shown that RL can tolerate a certain degree of reward inaccuracy. Moreover, RL tends to generalize better than SFT, which often relies on memorizing training data [6]. In RL, rewards are typically vague and serve primarily as directional signals for exploration, leading to RL's robustness to reward noise [34]. **(ii)** Prior work has also examined what constitutes a good reward model from an optimization perspective, revealing that more accurate reward models are not necessarily better teachers [45]. Therefore, reward signals estimated by the policy model itself may offer more suitable guidance for learning.

Online Learning. **TTRL** is designed based on an online RL approach, whereas traditional self-training and test-time training methods operate in an offline manner. The online nature of **TTRL** enables the model to improve its capabilities during the application, which in turn leads to more accurate labels generated through voting. As a result, the quality of the supervision signal improves, allowing for truly sustainable self-evolution. As shown in Figure 6, this dynamic learning process leads to a complementary improvement of performance in both pass@1 and maj@n.

5 Related Works

5.1 Test-Time Scaling

Test-Time Scaling (TTS) is designed to enhance the capabilities of Large Language Models (LLMs) in handling complex tasks by increasing computational resources at test time. Prior research [25, 40]

indicates that TTS is more efficient than scaling during pre-training [19]. Therefore, reallocating the same computational resources from pre-training to test-time could yield greater improvements in model performance. Current studies on TTS fall into two categories [51]: parallel generation and sequential generation. Parallel generation involves LLMs producing multiple candidate responses (self-consistency [4, 48], best-of-N [30, 41]), decision steps (Monte Carlo Tree Search [52, 63]), or tokens (Reward-guided Search [9, 20]) during inference. Subsequently, an aggregation strategy is applied to integrate these candidates, commonly using process reward models [24, 46, 60]. Concurrently, sequential generation focuses on extending the LLMs' output to include longer responses with reflective and chain-of-thought (CoT) processes [27, 49]. Although prompting techniques are widely adopted, they are often constrained by the capabilities of the underlying models. Notably, DeepSeek-R1 [11] is a representative advancement in this area, achieving extended reasoning capabilities in pre-trained language models through outcome-based reinforcement learning (RL), more specifically group relative policy optimization [38]. Compared to the first approach, which requires intensive process-level supervision [57], the second approach is more scalable due to its reliance on rule-based rewards.

Beyond the aforementioned methods that focus on scaling test-time inference computation, another approach to increasing test-time computing is **Test-Time Training (TTT)**. We introduce the relationship between these terminologies in Appendix F. While prior work has primarily focused on applications such as video generation and understanding [8, 12], and to some extent on large language models [1, 47], the integration of test-time scaling with reinforcement learning remains largely underexplored.

5.2 RL for Reasoning

Reinforcement Learning (RL) [44] plays a critical role in enhancing the instruction-following capabilities of Large Language Models (LLMs), particularly through approaches like Reinforcement Learning from Human Feedback (RLHF) [31]. RLHF aligns base models with human preferences using algorithms such as Proximal Policy Optimization (PPO) [36], where preference modeling is essential. Recently, Large Reasoning Models (LRMs), such as DeepSeek-R1 [11], have demonstrated the significance of RL in improving reasoning abilities using rule-based rewards, as exemplified by GRPO [38]. Unlike RLHF, which is tailored to open-domain instructions, GRPO is specifically designed to elicit long CoT [49] reasoning in mathematical problem-solving. Recent studies have focused primarily on improving the training stability of rule-based RL methods like GRPO and PPO [7, 26, 56]. However, these methods typically train LLMs only on supervised training data, while inference involves generating extended CoT reasoning on unseen test problems. Moreover, current RL approaches [15, 50] depend on verifiable outputs, such as solutions in mathematics or code, that can provide reliable reward signals.

Previous studies have explored self-rewarding [32, 58] and self-play training [5] for unlabeled data. However, these works primarily focus on open-domain instruction following [5, 58] rather than mathematical reasoning or employ preference-based optimization strategies [32] such as DPO [33] instead of online reinforcement learning algorithms. In addition to these studies, we identified several concurrent works [53, 61, 62], that explore self-supervised and semi-supervised reasoning using reinforcement-like methods. The key distinction lies in reward estimation: we employ majority voting, which is derived from the model itself and mitigates reward hacking. We acknowledge that future research integrating the insights and strengths of these approaches could lead to more robust reasoning models at the era of experience [39]. **TTRL offers a preliminary attempt at RL with self-labeled rewards, advancing toward learning from streams of experience.**

6 Conclusion

In this paper, we propose Test-Time Reinforcement Learning (**TTRL**), a novel framework for training large language models with Reinforcement Learning (RL) on test data without access to ground-truth labels. A key component of **TTRL** is its majority voting reward function, which generates rule-based rewards based on consensus among model predictions. Our experiments demonstrate the strong potential of **TTRL**, achieving consistent improvements across a variety of models and tasks. We view **TTRL** as a preliminary step toward RL with self-labeled rewards, marking an important direction of learning from continuous streams of experience.

Acknowledgments and Disclosure of Funding

This work is supported by National Science and Technology Major Project (2023ZD0121403), Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), National Natural Science Foundation of China (No. 62406165), and Shanghai Municipal Science and Technology Major Project. We thank anonymous reviewers for their insightful comments and suggestions.

References

[1] Ekin Akyürek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning. *arXiv preprint arXiv:2411.07279*, 2024.

[2] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

[5] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

[6] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

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

[8] Karan Dalal, Daniel Koceja, Gashon Hussein, Jiarui Xu, Yue Zhao, Youjin Song, Shihao Han, Ka Chun Cheung, Jan Kautz, Carlos Guestrin, et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.

[9] Haikang Deng and Colin Raffel. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. *arXiv preprint arXiv:2310.09520*, 2023.

[10] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

[12] Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models, 2024. URL https://arxiv.org/abs/2305.18466.

[13] Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e4f51680, 2025. Notion Blog.

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

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

[16] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL https://arxiv.org/abs/2503.24290.

[17] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

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

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[20] Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*, 2024.

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

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

[23] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

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

[25] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

[26] 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. *arXiv preprint arXiv:2503.20783*, 2025.

[27] 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.

[28] Ministral-8B-Instruct. Ministral-8b-instruct, 2024. URL https://mistral.ai/news/ministraux.

[29] MistralAI-NeMo. Mistralai-nemo, 2024. URL https://mistral.ai/news/mistral-nemo.

[30] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[31] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[32] 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. *arXiv preprint arXiv:2411.04109*, 2024.

[33] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[34] Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. *arXiv preprint arXiv:2503.15477*, 2025.

[35] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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

[37] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

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

[39] David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.

[40] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[41] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

[42] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization. *Arxiv*, 2019.

[43] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

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

[45] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6202–6209, 2020.

[46] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

[47] Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A Efros, and Xiaolong Wang. Test-time training on video streams. *Journal of Machine Learning Research*, 26(9):1–29, 2025.

[48] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

[50] Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

[51] Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

[52] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[53] Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Qiushi Sun, Kanzhi Cheng, Junxian He, Jun Liu, and Zhiyong Wu. Genius: A generalizable and purely unsupervised self-training framework for advanced reasoning. *arXiv preprint arXiv:2504.08672*, 2025.

[54] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[55] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

[57] Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

[58] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

[59] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

[60] Kaiyan Zhang, Jiayuan Zhang, Haoxin Li, Xuekai Zhu, Ermo Hua, Xingtai Lv, Ning Ding, Biqing Qi, and Bowen Zhou. Openprm: Building open-domain process-based reward models with preference trees. In *The Thirteenth International Conference on Learning Representations*, 2025.

[61] Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025.

[62] 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, 2025. URL https://arxiv.org/abs/2505.03335.

[63] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Section 3

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Conduct multiple experiments to calculate the pass@1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 3

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the research conducted in this paper conforms to all aspects of the NeurIPS Code of Ethics. Specifically, we ensured compliance regarding transparency, reproducibility, potential societal impacts, respect for privacy, and fairness. No ethical concerns arose from our methods, datasets, or experiments throughout the research process.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This paper presents foundational research and is not tied to any particular application.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Section 3

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Limitations and Future Works

Limitations This work represents an initial exploration of test-time reinforcement learning using self-labeled rewards. While our experimental results are promising, several aspects require further investigation. In particular, we plan to conduct a more in-depth analysis of the impact of prior knowledge and hyperparameter configurations, both of which play critical roles in reinforcement learning dynamics. We will provide comprehensive discussions and ablation studies in future revisions of this paper.

Future Works Building on our findings, we identify several directions for future research:

- **Theoretical Analysis:** Developing a formal convergence analysis of **TTRL**, particularly focusing on its ability to optimize toward the two upper bounds in § 4.1.
- **Online Learning with Streaming Data:** Extending **TTRL** to real-time learning scenarios, where models interact with continuously arriving data and adapt dynamically, that is Test-Time Adaptation [23].
- **Large-Scale Self-Supervised RL Training:** Scaling up **TTRL** to massive datasets and models to explore its potential in self-supervised regimes without human-labeled data.
- **Agentic Tasks and Scientific Discovery:** Applying **TTRL** to more complex, open-ended domains such as agentic tasks and multi-step scientific reasoning.

B When Might TTRL Fail?

At the algorithmic level, **TTRL** is not fundamentally different from existing RL algorithms and therefore inherits several of their characteristics, such as sensitivity to data difficulty, strong reliance on priors, and risk of collapse under certain conditions. At the implementation level, these issues are further amplified by the constraints of **TTRL**, which estimates labels via majority voting and operates exclusively on test data that is both sparse and previously unseen, potentially resulting in failures in certain scenarios. In our preliminary experiments, we identified two potential issues:

Inappropriate RL Hyperparameters. Hyperparameter settings play a crucial role in RL training, varying across projects and often leading to training failures. The influence of hyperparameters is further amplified in **TTRL** due to potential noise in reward estimation and the characteristics of the test data. Figure 11 presents a comparison of several unsuccessful attempts on AIME 2024. Both of these failed attempts exhibit persistently high entropy that does not diminish throughout training, consistent with findings of prior work [13]. In our preliminary experiments, we identified two key hyperparameters that can critically affect training stability and success:

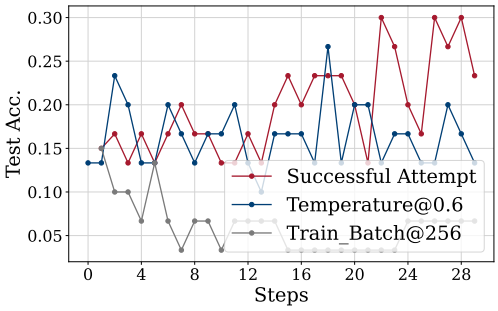


Figure 11: Failed attempts. We compare the curves under settings with appropriate parameters versus those with suboptimal ones.

- **Temperature:** Setting the temperature to 1.0, as opposed to 0.6, increases the model's output entropy. This promotes more extensive exploration and allows the model to make better use of its prior knowledge for self-improvement, which is particularly important when addressing challenging benchmarks.
- **Episodes:** Given the substantial variation in size and difficulty across datasets, smaller and more difficult datasets need more episodes to achieve sufficient exploration.

Lack of Prior Knowledge on Target Task. Prior knowledge plays a crucial role in RL, often determining the success or failure of the **TTRL** learning process. This is mainly because the test data generally exhibits higher difficulty and introduces new features, but **TTRL** does not incorporate mechanisms such as data filtering to support curriculum learning.

Table 3: Performance of **TTRL** across the five difficulty levels of MATH-500.

Metric	Name	MATH-500-L1	MATH-500-L2	MATH-500-L3	MATH-500-L4	MATH-500-L5
Accuracy	Backbone	25.9	33.0	36.3	32.5	22.3
	w/ **TTRL**	71.2	76.2	76.3	58.7	39.2
	Δ	+45.4 $\uparrow$ 175.3%	+43.2 $\uparrow$ 130.8%	+40.0 $\uparrow$ 110.2%	+26.2 $\uparrow$ 80.4%	+16.8 $\uparrow$ 75.3%
Response Len.	Backbone	2,339.2	2,125.1	2,120.6	1,775.1	1,751.3
	w/ **TTRL**	624.3	614.4	672.3	783.5	985.3
	Δ	−1,715.0 $\downarrow$ 73.3%	−1,510.6 $\downarrow$ 71.1%	−1,448.3 $\downarrow$ 68.3%	−991.6 $\downarrow$ 55.9%	−766.0 $\downarrow$ 43.7%

Therefore, for the same backbone, **TTRL** fails if the model's prior knowledge is insufficient to handle the complexity of the data. To further validate this hypothesis, we conduct an ablation study on MATH-500. We divide MATH-500 into five subsets according to its annotated difficulty levels, ranging from 1 to 5, and apply **TTRL** to each subset independently, using `Qwen2.5-Math-1.5B`. We then compare the results to those of the backbone, as shown in Table 3. We observe that as the question difficulty increases, both the performance improvement and length reduction ratios tend to decrease. **This suggests that the available prior knowledge of the backbone is insufficient to support learning on more challenging questions.**

C Reward Function Pseudo-Code

Listing 1: The pseudo-code of the majority voting reward function.

```python
from collections import Counter

def majority_voting_reward_fn(outputs):
    """
    Assigns a reward of 1 to each output whose extracted answer
        matches the majority answer, otherwise 0.
    """
    # Extract answers from each output
    answers = [extract_answer(output) for output in outputs]

    # Find the majority answer
    counts = Counter(answers)
    majority_answer, _ = counts.most_common(1)[0]

    # Assign rewards: 1 if matches majority, else 0
    rewards = [1 if ans == majority_answer else 0 for ans in answers]
    return rewards

outputs = llm.generate(problem, n=N)
rewards = majority_voting_reward_fn(outputs)
```

D Additional Results

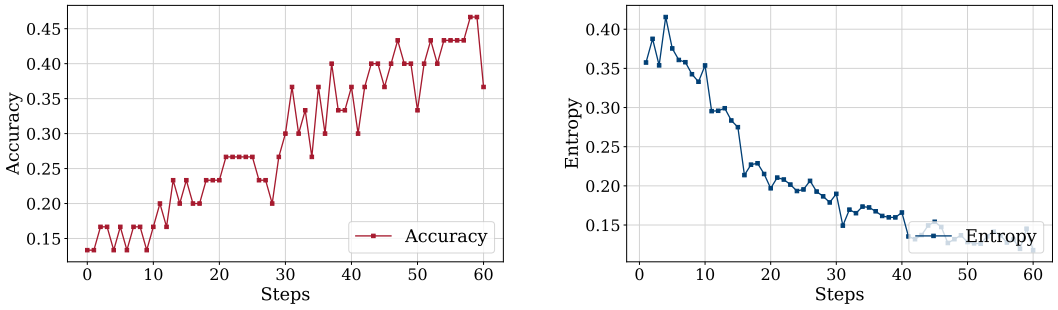
Table 4 shows pass@1 results using greedy decoding. For the two base models, we further include comparisons with their instruct versions that have undergone large-scale post-training. In addition, we include for reference current leading "R1-Zero-Like" models with similar backbones, which are extensively trained using RL: DeepSeek-R1-Distill-1.5B&7B [11], SimpleRL-Zero-7B [59], PRIME-Zero-7B [7], OpenReasoner-Zero-7B [16], Oat-Zero-1.5B&7B [26], and LIMR [22]. **Note that TTRL has a different setup from the previous models, which makes the comparison seem unfair.**

On the highly challenging mathematical reasoning benchmark AIME 2024, **TTRL** achieves a substantial improvement of 159.3%, surpassing all models trained on large-scale datasets. Furthermore,

when applied to `Qwen2.5-Math-7B`, **TTRL** yields an average improvement of 84.1% across three benchmarks. Figure 12 shows two curves of **TTRL** on AIME 2024 with `Qwen2.5-Math-7B` as an example.

Table 4: Additional results of **TTRL** on each task. * indicates results from Dr. GRPO [26]. Our training data size matches the corresponding benchmark dataset size.

Name	AIME 2024	AMC	MATH-500	Avg	Labeled Data
`Qwen2.5-Math-1.5B`*	20.0	32.5	33.0	28.5	-
w/ TTRL	20.0	53.0	**80.0**	**51.0**	✗
Δ	0	+20.5	+47.0	+22.5	✗
	0	↑ 63.1%	↑ 142.4%	↑ 79.0%	✗
Qwen2.5-Math-1.5B-Instruct*	10.0	48.2	74.2	44.1	3.1M
DeepSeek-R1-Distill-1.5B@3k*	2.5	21.7	52.2	25.5	800K
DeepSeek-R1-Distill-1.5B@8k*	20.0	49.4	77.4	48.9	800K
Oat-Zero-1.5B*	**20.0**	**53.0**	74.2	49.1	8.9K
`Qwen2.5-Math-7B`*	16.7	38.6	50.6	35.3	-
w/ TTRL	43.3	**67.5**	**84.2**	**65.0**	✗
Δ	+26.6	+28.9	+33.6	+29.7	✗
	↑ 159.3%	↑ 74.9%	↑ 66.4%	↑ 84.1%	✗
Qwen2.5-Math-7B-Instruct*	16.7	53.0	83.6	51.1	3.1M
DeepSeek-R1-Distill-7B@3k*	10.0	26.2	60.1	32.1	800K
SimpleRL-Zero-7B*	26.7	60.2	78.2	55.0	8.9K
PRIME-Zero-7B*	16.7	62.7	83.8	54.4	230K
OpenReasoner-Zero-7B@3k*	13.3	47.0	79.2	46.5	129K
Oat-Zero-7B*	**43.3**	62.7	80.0	62.0	8.9K
LIMR-7B	32.5	63.8	78.0	58.1	1.4K



(a) Accuracy Curve.

(b) Entropy Curve.

Figure 12: The entropy and accuracy curves of **TTRL** on AIME 2024 with `Qwen2.5-Math-7B`.

E Training Metrics

Given the absence of ground-truth labels in the test data, evaluating the performance of **TTRL** throughout the training process presents a challenge. To mitigate this limitation, we introduce a set of training-time metrics specifically designed to monitor and assess the effectiveness of **TTRL**. These metrics inform the selection of the optimal checkpoint and provide valuable insights regarding training dynamics.

- **Entropy**: Measures the uncertainty of the model's generation.
- **Majority Voting Reward**: Rule-based rewards computed from the majority-voted label.
- **Majority Ratio**: The frequency of the most common answer within a rollout.

Furthermore, we define several metrics that rely on access to ground-truth labels, which allow for a deeper analysis of the model's behavior during training:

- **Label Accuracy (maj@n)**: Indicates whether the estimated label matches ground-truth.
- **Reward Accuracy**: Indicates the proportion of majority voting rewards (computed from the estimated label) that match rewards computed from the ground-truth label.
- **Ground-Truth Ratio**: The frequency of the ground-truth answer within a rollout.

F Terminology

Test-time scaling refers to increasing computational resources during test time, which can be categorized into test-time training and test-time inference. These two approaches are complementary. We will provide an introduction below.

Table 5: Terminology relationship.

Name	Category	Methods
Test-Time Scaling (TTS)	Test-Time Training (TTT)	Test-Time Reinforcement Learning (TTRL)
	Test-Time Inference (TTI)	Majority Voting, Best-of-N

F.1 Test-Time Training (TTT)

Test-Time Training (TTT) is a technique for adapting a pre-trained model at inference time to improve generalization under distribution shifts. Let f_θ denote a model trained on a source domain $\mathcal{D}s = \{(x_i, y_i)\}i = 1^N$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and θ represents the learned parameters. During standard inference, the model is evaluated on test samples $x_t \sim \mathcal{D}_t$ with fixed parameters θ, where $\mathcal{D}_t \neq \mathcal{D}_s$.

In contrast, TTT allows the model to adapt to each test sample x_t by minimizing an auxiliary self-supervised loss $\mathcal{L}_{\text{aux}}$, without access to labels y_t. The model parameters are updated online with the auxiliary task, which is typically designed to be label-free and consistent with the main task.

F.2 Test-Time Inference (TTI)

Test-Time Inference (TTI) refers to the strategy of enhancing the performance of a large language model during inference by allocating additional computational resources. Formally, let f_θ denote a language model with parameters θ, and let x be an input prompt. The model generates an output y by sampling from the conditional distribution $p_\theta(y \mid x)$. TTI techniques aim to improve the quality of y by employing methods such as generating multiple candidate outputs and selecting the best one based on a scoring function, or by refining the output through iterative processes [51].

One common approach involves generating N candidate outputs $\{y_1, y_2, \ldots, y_N\}$ and selecting the optimal output y^* using a scoring function $s(y, x)$:

$$y^* = \arg\max_{y_i} s(y_i, x) \tag{4}$$

The scoring function $s(y, x)$ can be instantiated in various ways, such as:

1. Majority Voting (MV): Selecting the most frequent output among the candidates.
2. Best-of-N (BoN): Using reward models to score each candidate, then selecting the highest-scoring one.
3. Weighted BoN: Integrating MV and BoN strategies to leverage their respective strengths.