M-GRPO: Stabilizing Self-Supervised Reinforcement Learning for Large Language Models with Momentum-Anchored Policy Optimization

Bizhe Bai[1,2], **Hongming Wu** [2], **Peng Ye**[3,4], **Tao Chen**[1,2],

[1]Shanghai Innovation Institute
[2]College of Future Information Technology, Fudan.
[3] Shanghai AI Laboratory
[4] The Chinese University of Hong Kong
bizhe.bai@sii.edu.cn,eetchen@fudan.edu.cn

Abstract

Self-supervised reinforcement learning (RL) presents a promising approach for enhancing the reasoning capabilities of Large Language Models (LLMs) without reliance on expensive human-annotated data. However, we find that existing methods suffer from a critical failure mode under long-horizon training: a "policy collapse" where performance precipitously degrades. We diagnose this instability and demonstrate that simply scaling the number of rollouts—a common strategy to improve performance—only delays, but does not prevent, this collapse. To counteract this instability, we first introduce M-GRPO (Momentum-Anchored Group Relative Policy Optimization), a framework that leverages a slowly evolving momentum model to provide a stable training target. In addition, we identify that this process is often accompanied by a rapid collapse in policy entropy, resulting in a prematurely confident and suboptimal policy. To specifically address this issue, we propose a second contribution: an adaptive filtering method based on the interquartile range (IQR) that dynamically prunes low-entropy trajectories, preserving essential policy diversity. Our extensive experiments on multiple reasoning benchmarks demonstrate that M-GRPO stabilizes the training process while the IQR filter prevents premature convergence. The combination of these two innovations leads to superior training stability and state-of-the-art performance. Code is available at M_GRPO.

1 Introduction

Reinforcement learning with verifiable reward (RLVR) has become a central ingredient in post-training large language models (LLMs) for complex reasoning tasks (1; 2). While RLVR can substantially improve helpfulness and reliability, it is expensive and domain-limited because it requires large amounts of carefully curated human preference data and reward modeling infrastructure. Recent work therefore explores self-supervised or label-free RL signals for reasoning: leveraging a model's own uncertainty or self-consistency to synthesize rewards and thereby train without ground-truth answers or programmatic verifiers. They train on unlabeled prompts by constructing intrinsic signals from the model itself, such as self-consistency–derived correctness proxies or self-certainty as a reward (3; 4). Some other methods include test-time reinforcement that turns majority-vote statistics into a usable objective such as SRT (5) and TTRL (6).

Self-supervised RLVR are easy to collapse These Self-supervised RLVR(SS-RLVR) approaches report promising early gains, but we find they suffer a critical failure mode under long-horizon

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

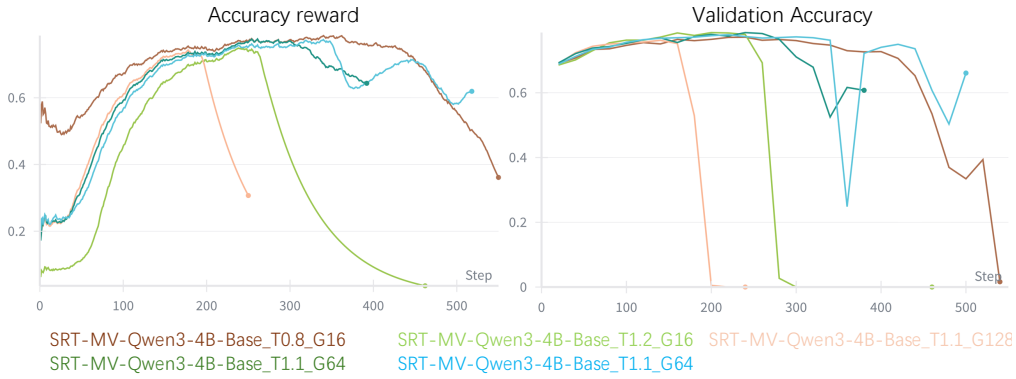

Figure 1: This caption details the reproduction of the SRT method (5). The experiments were conducted with different numbers of rollouts and a set temperature. For all methods, training was performed on the train split of the MATH dataset (7) without access to ground truth answers, using a consistent batch size and learning rate. The models were then validated on the corresponding MATH test split. Further test results, derived from the best-performing checkpoint chosen manually, are available in Table 1.

training: the policy collapses. In our reproduction of SRT (5) and Intuitor (3) self-supervised training on MATH dataset (7) as shown in (Fig. 5), the training reward rises initially and then precipitously or progressively crashes , accompanied by degradation in validation accuracy on the test set. This finding also aligns with the "model collapse" scenario with Sheikh's (5) and Zizhuo's (4).

Scaling Rollouts Delays, But Does Not Prevent, Policy Collapse While increasing the number of rollouts during training is a known technique to bolster performance in supervised RLVR, as noted by recent work (8; 9), its application in the self-supervised context reveals a significant challenge. We investigate this scaling behavior and find a precarious trade-off: although more rollouts can improve the model's best performance (Table 1), they only serve to slow the rate of descent into policy collapse, rather than preventing it entirely (Fig. 1). This phenomenon suggests that while scaling the evidence pool for the self-rewarding mechanism can temporarily mitigate inherent instability, the model ultimately succumbs to the identified failure mode. This finding underscores the need for a stabilization method that can harness the benefits of larger rollout batches without eventually facing performance degradation.

	Math			Reasoning	
	MATH500	**AIME24**	**AIME25**	**GPQA Diamond**	**GPQA**
			Qwen3-4B-base		
Original	0.615	.00833	0.0500	0.3441	0.2991
SRT-T1.1-G16	0.7515	0.0917	0.1042	0.3706	0.3482
SRT-T0.8-G16	0.7395	0.0958	0.0708	0.3655	0.3772
SRT-T1.2-G16	0.7615	0.1042	0.0708	0.3649	0.3571
SRT-T1.1-G32	0.7405	0.1083	0.1062	0.3586	0.3348
SRT-T1.1-G64	0.792	0.1250	0.1167	0.3826	0.3504
SRT-T1.1-G128	0.762	0.0917	0.0875	0.3586	0.3638

Table 1: Scaling result of rollout numbers G. Results with different rollout numbers and temperatures for SRT (5). T denotes the sampling temperature and G the total number of rollouts (e.g., `SRT-T0.8-G16` means temperature $T = 0.8$ and $G = 16$ rollouts). Although SRT improves at higher G, it remains prone to collapse as shown in Fig. 1. All results are obtained by manually selecting the best checkpoint based on accuracy reward before collapse.

Self supervised reinforcement learning also caused entropy collapse The phenomenon of policy entropy collapse (10) is also prevalent in self-supervised reinforcement learning, as illustrated on the

left of Fig. 2. During the initial stages of training, the policy entropy declines precipitously, resulting in a prematurely confident policy model.

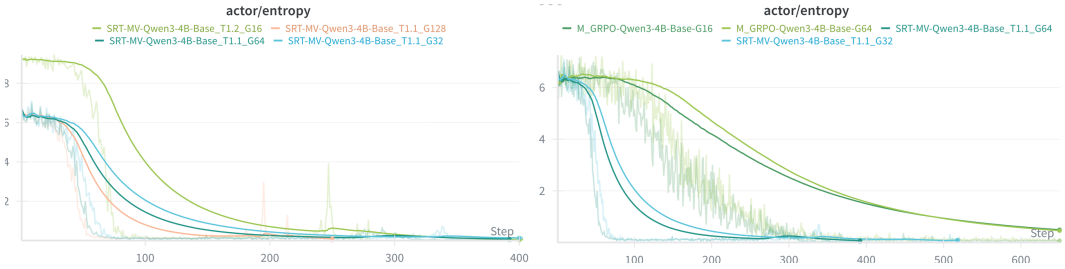

Figure 2: **Left:** The evolution of policy entropy during the training of a standard SRT model (6; 5). A sharp drop in entropy is observed early in training, leading to an overly confident policy. **Right:** A comparison with our proposed method, initialized with M_GRPO. Our approach maintains a higher level of entropy that decreases more gradually throughout the training process, mitigating premature convergence.

Contributions. First, we diagnose "policy collapse," a critical instability in self-supervised reinforcement learning for LLMs, and show that simply scaling rollouts exacerbates this failure. Second, we introduce M-GRPO, a novel momentum-anchored framework that leverages a slowly evolving model to provide a stable training target, effectively mitigating said collapse. Third, to prevent policy entropy collapse, we propose a filtering method based on interquartile range that adaptively filters the low-entropy trajectories. Finally, we demonstrate through extensive experiments that M-GRPO achieves superior stability and state-of-the-art performance on multiple reasoning benchmarks.

2 Methodology

This chapter is structured to provide a clear and comprehensive overview of our proposed methodology. We begin in Section 2.1 with a detailed description of the Momentum-anchored self-supervised reinforcement learning model. The subsequent section, Section 2.2, is dedicated to explaining the interquartile range (IQR) based method for trajectory entropy filtering. Finally, the complete, integrated algorithm is presented in Section 2.3.

2.1 Momentum-anchored self-supervised RL

To enhance the stability of the self-supervised process in policy refinement, we introduce a momentum-based RLVR framework, M-GRPO. The main framework is illustrated in Fig. 3 Our approach is built upon the foundational principles of Group Relative Policy Optimization (GRPO) (1; 2), but it incorporates a momentum-updated model to provide a more consistent and reliable training signal. This method stabilized through the momentum mechanism. The core idea is inspired by the success of momentum contrast in self-supervised visual representation learning (11; 12), which we adapt to the context of policy optimization for self-supervised RLVR.

Formally, our framework consists of two models: the current policy model π_{θ_q} , which we aim to train, and a momentum-based model π_{θ_k}. The parameters θ_k of the momentum model are not updated by backpropagation. Instead, they are an exponential moving average of the current policy model's parameters θ_q. Following each training iteration of the current policy model, the momentum-model parameters are updated as follows:

$$\pi_{\theta_k} \leftarrow m\pi_{\theta_k} + (1-m)\pi_{\theta_q}, \tag{1}$$

where $m \in [0, 1)$ is a momentum coefficient. A high value of m (e.g., 0.99) ensures that the momentum model evolves slowly, thereby providing a stable source for generating reference outputs.

For each input prompt x in a given training batch, we generate two distinct sets of rollouts. The current policy model policy $\pi_{\theta_q}(Y|x)$ samples M responses, $\{y_i^q\}_{i=1}^M$, while the momentum policy model $\pi_{\theta_k}(Y|x)$ samples N responses, $\{y_j^k\}_{j=1}^N$. These are combined to form a pool of $G = M + N$ total rollouts.

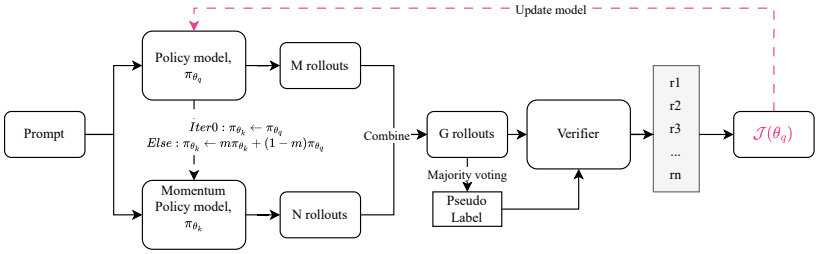

Figure 3: An illustration of our proposed momentum-based policy refinement framework. The policy π_{θ_q} is trained using feedback derived from rollouts generated by both itself and a slowly evolving momentum policy π_{θ_k}. The momentum's parameters are a moving average of the current policy model's, providing a stable target for pseudo-label generation via majority voting.

From this combined pool, we identify a pseudo-ground truth response, y_v, using a majority voting mechanism. This mechanism selects the response that has the highest agreement with all other responses in the pool: $y_v \leftarrow \arg\max_{y^* \in Y_{\text{pool}}} \sum_{y' \in Y_{\text{pool}}} \mathbb{I}[\text{ans}(y') = \text{ans}(y^*)]$, where $\mathbb{I}[\cdot]$ is the indicator function and $\text{ans}(\cdot)$ extracts the final answer from a response. The inclusion of the stable momentum model's rollouts in the voting pool is crucial for mitigating the noise and instability of pseudo-labels generated purely from the rapidly changing current policy model .With the pseudo-ground truth y_v established, we can calculate the reward scores of the current policy model's M rollouts based on pseudo-ground truth y_v. The reward is binary: 1 for agreement and 0 for disagreement. Following the GRPO framework, we then compute the normalized advantage for each of the current policy model's sampled responses:

$$\hat{A}_i = \frac{r(y_v, y_i^q) - \text{mean}(\{r(y_v, y_j^q)\}_{j=1}^M)}{\text{std}(\{r(y_v, y_j^q)\}_{j=1}^M)}. \tag{2}$$

This advantage normalization is performed on a per-prompt basis over the current policy model's M rollouts and serves to reduce the variance of the policy gradient estimate.

The final learning objective for the current policy model π_{θ_q} is to maximize the expected advantage-weighted log-likelihood of its generated responses:

$$\mathcal{J}(\theta_q) = \mathbb{E}_{x \sim D, \{y_i^q\}_{i=1}^M \sim \pi_{\theta_q}(Y|x)} \left[\sum_{i=1}^{M} \hat{A}_i \log \pi_{\theta_q}(y_i^q|x) \right]. \tag{3}$$

2.2 IQR-based trajectory entropy filter

A critical challenge in self-supervised reinforcement learning is the management of trajectory quality. Trajectories with low entropy often correspond to policies that have prematurely converged or are overly confident, which can degrade the quality of the pseudo-labels used for training. To address this, we introduce a dynamic filtering mechanism based on the interquartile range (IQR) to prune low-entropy trajectories.

For each input sample x from a mini-batch $\mathcal{B}$, we generate a total of G rollouts, comprising M trajectories from the current policy π_{θ_q} and N trajectories from the momentum policy π_{θ_k}. For each of these G trajectories, we compute its trajectory-level entropy. The set of all trajectory entropies for a given input sample forms a distribution.

Rather than employing a static threshold (13), such as removing the bottom 10% of trajectories, which may be ill-suited for the dynamic nature of training, we utilize the IQR to identify and discard outliers. Specifically, for the set of G entropy values associated with an input sample, we calculate the first quartile (Q_1) and the third quartile (Q_3). The interquartile range is then defined as $\text{IQR} = Q_3 - Q_1$. A trajectory is identified as a low-entropy outlier if its entropy falls below the threshold $Q_1 - k \cdot \text{IQR}$, where k is a hyperparameter that controls the sensitivity of the filter. In our experiments, we set $k = 0.75$.

This dynamic approach is particularly advantageous because the distribution of trajectory entropies can vary significantly, both across different input samples and throughout the training process. For instance, at the beginning of training, most trajectories may exhibit high entropy as the policy explores the action space. A static filter would be ineffective in this scenario, whereas our IQR-based method

adapts to the local distribution of entropies, ensuring that only genuine outliers are removed. This leads to a more robust and stable training process by preserving high-quality, high-entropy trajectories for the subsequent learning steps.

2.3 Pseudo-code

The overall training procedure for our Momentum-anchored GRPO (M-GRPO) with IQR-based filtering is summarized in Algorithm 1. By leveraging a momentum policy model to stabilize the generation of pseudo-labels and a dynamic filtering mechanism to ensure their quality, our method provides a more consistent learning signal. This empirically leads to improved performance and training stability, effectively reducing the risk of policy degradation or collapse.

Algorithm 1 M-GRPO with IQR-based Trajectory Entropy Filtering

Input: policy model π_{θ_q}, training dataset $\mathcal{D}$, learning rate η, momentum coefficient m, number of policy model rollouts M, number of momentum rollouts N, total number of rollouts $G = M+N$, IQR coefficient $k = 0.75$.
Initialize momentum model parameters $\pi_{\theta_k} \leftarrow \pi_{\theta_q}$.
for each training iteration **do**
  Sample a mini-batch of prompts $\mathcal{B} \subseteq \mathcal{D}$.
  for each prompt $x \in \mathcal{B}$ **do**
    Generate M responses from the current policy: $\{y_i^q\}_{i=1}^M \sim \pi_{\theta_q}(\cdot|x)$.
    Generate N responses from the momentum model: $\{y_j^k\}_{j=1}^N \sim \pi_{\theta_k}(\cdot|x)$.
    Combine all generated responses: $\mathcal{Y} = \{y_i^q\}_{i=1}^M \cup \{y_j^k\}_{j=1}^N$.
    Calculate trajectory-level entropy for each response in $\mathcal{Y}$.
    Compute Q_1 and Q_3 for the set of trajectory entropies.
    Define the filtering threshold: $T_{IQR} = Q_1 - k \cdot (Q_3 - Q_1)$.
    Filter the set of responses: $\mathcal{Y}_{filtered} = \{y \in \mathcal{Y} \mid \text{entropy}(y) \geq T_{IQR}\}$.
    Identify the majority-voted pseudo-ground truth y_v from $\mathcal{Y}_{filtered}$.
    Estimate the relative advantages $\hat{A}_i$ for responses from the current policy model in $\mathcal{Y}_{filtered}$ using Eq. (2).
  end for
  Calculate the policy objective $\mathcal{J}(\theta_q)$ over the mini-batch using the filtered, high-quality trajectories and Eq. (3).
  Update current policy parameters: $\theta_q \leftarrow \theta_q + \eta \nabla_{\theta_q} \mathcal{J}(\theta_q)$.
  Update momentum model parameters: $\pi_{\theta_k} \leftarrow m\pi_{\theta_k} + (1-m)\pi_{\theta_q}$ using Eq. (1).
end for

3 Experiment

Backbone Models and Baselines. We utilize Qwen3-4B-Base (14) as LLM backbone. We compare our result with SRT (5).

Implementation Details We implement our algorithm on top of the VeRL framework (15). Experiments are conducted on 8 × NVIDIA H200 GPUs. Specifically, methods are trained on the training split of the MATH dataset without its provided ground truth (7). We test all models' accuracy on the test split of MATH dataset as well. For every RL update during training, we generate 32 candidate rollouts per problem and set temperature to 1.1 as default. We test model performance on MATH500 (7), AIME25 (16; 17), GPQA Diamond (18; 19), GPQA (20), and Livecode (21). For AIME problems and GPQA Diamond problems, we sample 8 times and take the average accuracy. For the rest of therest of the benchmarks, we sample once. For M-GRPO, momentum policy model rollouts N=$\frac{G}{4}$. For additional experimental details, please refer to Table 4.

4 Result

Our experimental results validate the effectiveness of M-GRPO in stabilizing the self-supervised reinforcement learning process and achieving superior performance compared to the baseline SRT method.

	Math			Reasoning		Code
	MATH500	**AIME24**	**AIME25**	**GPQA Dia**	**GPQA**	**LiveCode**
			Qwen3-4B-base			
Original	61.50%	0.83%	5.00%	34.41%	29.91%	9.61%
SRT_{Best}	79.20%	12.50%	11.67%	38.26%	35.04%	19.69%
SRT_{Final}	47.50%	7.50%	8.75%	28.54%	25.89%	16.12%
M-GRPO_{Final}	79.70%	12.50%	13.33%	37.94%	37.72%	27.12%
$\text{M-GRPO+IQR}_{Final}$	79.75%	14.58%	14.17%	39.65%	35.49%	\

Table 2: Main Results of RL performance comparison on reasoning benchmarks. Cell background colors indicate relative performance: darker colors denote better results within each model group. We compare M-GRPO with SRT (5) on Qwen3-4B-Base model trained on MATH (7) dataset without ground-truth label. SRT_{Best} means we manually chosen the best checkpoint based on accuracy reward. SRT_{Final} means the final step checkpoint. M-GRPO show superior performance compared to SRT according to best accuracy and stability.

4.1 M-GRPO Prevents Policy Collapse and Achieves State-of-the-Art Performance

A primary contribution of our work is the mitigation of "policy collapse," a critical failure mode in self-supervised RLVR. As illustrated in Fig 1 and further detailed in Fig5, the SRT (5) baseline exhibits a characteristic pattern of instability. This instability necessitates manually selecting the best checkpoint (`SRT-Best`) before the performance crash, which is impractical for continuous training and deployment.

In stark contrast, M-GRPO demonstrates remarkable training stability. As shown in Fig 4, M-GRPO sustains an improving stable reward throughout the training horizon, which translates directly to a consistently high validation accuracy without any signs of degradation. This stability obviates the need for manual intervention or checkpoint cherry-picking.

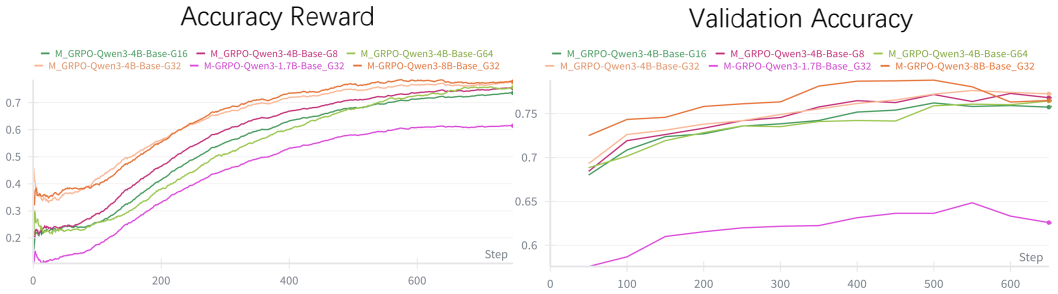

Figure 4: Training reward on the training split of **MATH** dataset and validation accuracy on test split with different training backbone including Qwen3-4B-Base,Qwen3-1.7B-Base and Qwen3-8B-Base. Our proposed M-GRPO sustains stable reward signals throughout training.

The quantitative results of this stability are presented in Table 2. At the final training checkpoint, `M-GRPO-Final` significantly outperforms `SRT-Final` across all six reasoning benchmarks, highlighting the severe performance loss SRT suffers from collapse. More importantly, `M-GRPO-Final` is highly competitive with, and frequently surpasses, the manually selected `SRT-Best`. For instance, on GPQA, M-GRPO achieves a final accuracy of 40.09%, compared to SRT's peak of 35.04%. The improvements are observed on more challenging benchmarks such as AIME24 (+2.92%), GPQA (+5.05%), and LiveCode (+7.43%) in absolute accuracy. These results confirm that M-GRPO not only stabilizes training but also enables the model to converge to a more robust and capable state.

4.2 Analysis of Rollout Scaling

To further understand the dynamics of self-supervised RLVR, we analyzed the impact of varying the number of rollouts (G) on the SRT baseline's performance and stability. As shown in Table 1,

	Math			Reasoning			Code
	MATH500	**AIME24**	**AIME25**	**GPQA Dia**	**GPQA**	**MMLU-pro**	**mbpp(22)**
			Qwen3-4B-base				
Original	61.50%	0.83%	5.00%	34.41%	29.91%	51.38%	63.40%
M-GRPO+IQR$_{G8}$	77.60%	11.25%	10.42%	39.02%	37.50%	56.05%	68.60%
M-GRPO+IQR$_{G16}$	79.75%	14.43%	10.00%	39.65%	33.94%	57.05%	70.40%
M-GRPO+IQR$_{G32}$	79.75%	14.58%	14.17%	39.65%	35.49%	55.47%	70.60%
M-GRPO+IQR$_{G256}$	79.50%	16.67%	14.17%	40.66%	38.39%	55.08%	70.40%

Table 3: Scaling analysis of the M-GRPO+IQR method on math, reasoning, and code generation benchmarks. This table illustrates the effect of increasing the number of rollouts (G) on model performance. A clear trend of improvement is observed as G is scaled from 8 to 32. However, the performance gains begin to plateau beyond this point, with only minimal improvements observed when increasing G to 256.

increasing the number of rollouts can indeed improve the peak performance of SRT. For example, on MATH500, scaling from G=16 to G=64 boosts the best-achieved accuracy from 76.15% to 79.20%. For M-GRPO, as illustrated in Table 3, a clear trend of improvement is observed as G is scaled from 8 to 32. However, the performance gains begin to plateau beyond this point, with only minimal improvements observed when increasing G to 256.

5 Conclusion

In this paper, we have identified a critical instability issue in self-supervised reinforcement learning for large language models, leading to policy collapse during extended training. We have demonstrated that this failure is attributable to the inherent lack of a stable target in self-rewarding systems. To address this, we introduced M-GRPO, a momentum-anchored, majority-voting framework that stabilizes the learning process.

References

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

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

[3] X. Zhao, Z. Kang, A. Feng, S. Levine, and D. Song, "Learning to reason without external rewards," 2025. [Online]. Available: https://arxiv.org/abs/2505.19590

[4] Z. Zhang, J. Zhu, X. Ge, Z. Zhao, Z. Zhou, X. Li, X. Feng, J. Yao, and B. Han, "Co-reward: Self-supervised reinforcement learning for large language model reasoning via contrastive agreement," 2025. [Online]. Available: https://arxiv.org/abs/2508.00410

[5] S. Shafayat, F. Tajwar, R. Salakhutdinov, J. Schneider, and A. Zanette, "Can large reasoning models self-train?" 2025. [Online]. Available: https://arxiv.org/abs/2505.21444

[6] Y. Zuo, K. Zhang, S. Qu, L. Sheng, X. Zhu, B. Qi, Y. Sun, G. Cui, N. Ding, and B. Zhou, "Ttrl: Test-time reinforcement learning," *ArXiv*, vol. abs/2504.16084, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:277993666

[7] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," 2021. [Online]. Available: https://arxiv.org/abs/2103.03874

[8] Z. Tan, H. Geng, M. Zhang, X. Yu, G. Wan, Y. Zhou, Q. He, X. Xue, H. Zhou, Y. Fan, Z. Li, Z. Zhang, G. Zhang, C. Zhang, Z. Yin, and L. Bai, "Scaling behaviors of llm reinforcement learning post-training: An empirical study in mathematical reasoning," 2025. [Online]. Available: https://arxiv.org/abs/2509.25300

[9] J. Hu, M. Liu, X. Lu, F. Wu, Z. Harchaoui, S. Diao, Y. Choi, P. Molchanov, J. Yang, J. Kautz, and Y. Dong, "Brorl: Scaling reinforcement learning via broadened exploration," 2025. [Online]. Available: https://arxiv.org/abs/2510.01180

[10] G. Cui, Y. Zhang, J. Chen, L. Yuan, Z. Wang, Y. Zuo, H. Li, Y. Fan, H. Chen, W. Chen, Z. Liu, H. Peng, L. Bai, W. Ouyang, Y. Cheng, B. Zhou, and N. Ding, "The entropy mechanism of reinforcement learning for reasoning language models," 2025. [Online]. Available: https://arxiv.org/abs/2505.22617

[11] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," 2020. [Online]. Available: https://arxiv.org/abs/2003.04297

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020. [Online]. Available: https://arxiv.org/abs/1911.05722

[13] X. Zhang, S. Wen, W. Wu, and L. Huang, "Edge-grpo: Entropy-driven grpo with guided error correction for advantage diversity," 2025. [Online]. Available: https://arxiv.org/abs/2507.21848

[14] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu, "Qwen3 technical report," 2025. [Online]. Available: https://arxiv.org/abs/2505.09388

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

[16] "Aime. aime problems and solutions,," 2025. [Online]. Available: https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions

[17] "Aime. aime problems and solutions,," 2025. [Online]. Available: https://huggingface.co/datasets/Maxwell-Jia/AIME_2024

[18] "Gpqa diamond," 2025. [Online]. Available: https://epoch.ai/benchmarks/gpqa-diamond

[19] "Gpqa diamond," 2025. [Online]. Available: https://huggingface.co/datasets/fingertap/GPQA-Diamond

[20] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman, "Gpqa: A graduate-level google-proof qa benchmark," 2023. [Online]. Available: https://arxiv.org/abs/2311.12022

[21] N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica, "Livecodebench: Holistic and contamination free evaluation of large language models for code," 2024. [Online]. Available: https://arxiv.org/abs/2403.07974

[22] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton, "Program synthesis with large language models," 2021. [Online]. Available: https://arxiv.org/abs/2108.07732

A Appendix

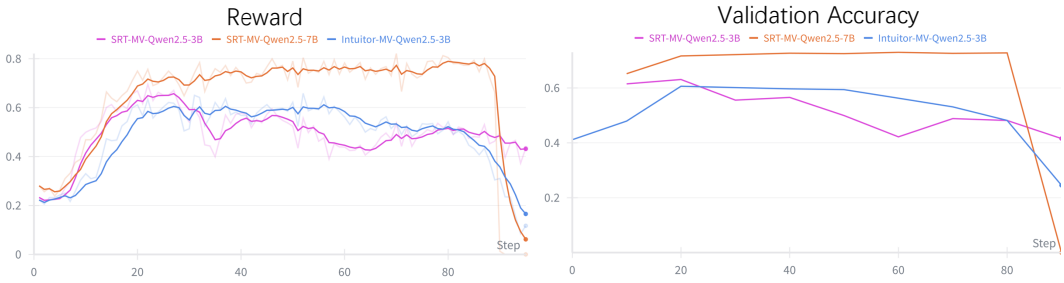

Figure 5: Reproduction of SRT-MV-Qwen2.5-3B, SRT-MV-Qwen2.5-7B based on SRT (5), and Intuitor-MV-Qwen2.5-3B based on Intuitor (3), which all self-supervised training on train split of MATH dataset (7) and validate on test split of MATH dataset. The training reward rises initially and then precipitously or progressively crashes , accompanied by degradation in validation accuracy on the test set.

A.1 Algorithm pipeline pseudo code

A.2 More experiment details

Table 4: More detailed experimental parameter settings.

Training Configuration	
Train Batch Size (Number of Sampled Questions)	8
Max Prompt Length	512
Max Response Length	3072
Clip Ratio	0.2
Optimizer Parameters	
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
Learning Rate	1e-06
Warmup Style	Cosine
Warmup Steps Ratio	0.1
KL Loss Coefficient	0.005
Temperature	
Training Temperature	1.1
Evaluation Temperature	0.8