Enhancing Efficiency and Exploration in Reinforcement Learning for LLMs

Mengqi Liao[1,2], Xiangyu Xi[2], Ruinian Chen[2], Jia Leng[2], Yangen Hu[2], Ke Zeng[2], Shuai Liu[1], Huaiyu Wan [1,3]*

[1]School of Computer Science and Technology, Beijing Jiaotong University
[2]Meituan
[3]Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence
***Correspondence:** hywan@bjtu.edu.cn

Abstract

Reasoning large language models (LLMs) excel in complex tasks, which has drawn significant attention to reinforcement learning (RL) for LLMs. However, existing approaches allocate an equal number of rollouts to all questions during the RL process, which is inefficient. This inefficiency stems from the fact that training on simple questions yields limited gains, whereas more rollouts are needed for challenging questions to sample correct answers. Furthermore, while RL improves response precision, it limits the model's exploration ability, potentially resulting in a performance cap below that of the base model prior to RL. To address these issues, we propose a mechanism for dynamically allocating rollout budgets based on the difficulty of the problems, enabling more efficient RL training. Additionally, we introduce an adaptive dynamic temperature adjustment strategy to maintain the entropy at a stable level, thereby encouraging sufficient exploration. This enables LLMs to improve response precision while preserving their exploratory ability to uncover potential correct pathways. The code and data is available on: https://github.com/LiaoMengqi/E3-RL4LLMs

1 Introduction

Large language models (LLMs) have gained considerable attention for their capabilities across a wide range of applications (Kumar, 2024). Recently, advanced reasoning models trained with reinforcement learning (RL) , such as DeepSeek-R1 (Guo et al., 2025) and Kimi k1.5, (Team et al., 2025) have demonstrated remarkable improvements in complex tasks like mathematics and coding, further intensifying research interest in RL for LLMs. After that, many works related to reinforcement learning for LLMs emerged (Xu et al., 2025; Yu et al., 2025; Liu et al., 2025). Among them, the most common combination is to use the GRPO (Shao et al., 2024) algorithm or its variants combined with rule-based rewards for reinforcement learning.

However, rule-based rewards result in very sparse reward signals. When the training data is particularly challenging, the policy struggles to sample the correct answer, causing the advantage in the GRPO algorithm to become zero. In such cases, the policy fails to obtain an update gradient. DAPO (Yu et al., 2025) filters out samples with a within-group reward standard deviation of zero to avoid zero advantage and employs multiple rounds of online sampling until enough experience is gathered for a single update. This approach is highly inefficient because the cost of sampling experiences is extremely high, yet this method results in a substantial waste of rollouts.

What's more, Yue et al. (2025) highlights that during evaluation, while the RL model surpasses the base model with a small sample size (small k), the base model achieves superior pass@k performance as the sample size increases (large k). This occurs because reinforcement learning prioritizes maximizing rewards, leading the model to focus probabilities on high-reward paths, potentially neglecting diverse correct answers. While encouraging more exploration during training could potentially mitigate this issue, our experiments reveal that combining entropy regularization (Mnih et al., 2016; Williams, 1992)—the widely adopted approach for fostering exploration in deep reinforcement learning—with sparse rule-based rewards can degrade performance, especially when training with challenging questions, and may even result in model collapse.

To address the aforementioned issues, we first introduce a dynamic rollout budget allocation mechanism to enhance the training efficiency of RL. For simple questions that the model can answer proficiently, we reduce their rollout budget, as performing reinforcement learning on such problems yields minimal gains. The saved rollout budget is

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 1451–1463
November 4-9, 2025

reallocated to more challenging problems, thereby increasing the likelihood of obtaining correct answers. Additionally, to promote exploration without introducing harmful gradients, we propose a temperature scheduler that dynamically adjusts the temperature to maintain a stable policy entropy level, thereby enabling more extensive exploration during training. An annealing mechanism is further integrated to effectively balance exploration and exploitation. In summary, our contributions are as follows:

- We propose a dynamic rollout budget allocation mechanism that enables a more rational distribution of computational resources, allowing RL to be conducted more efficiently.
- We introduce a temperature scheduler that dynamically adjusts the sampling distribution's temperature, maintaining the entropy at a stable level to encourage more exploration.
- Experimental results demonstrate that our method improves the 7B model's pass@1 by 5.31% and pass@16 by 3.33% on the AIME 2024 benchmark compared to train with GRPO only, and consistently outperforms GRPO in pass@16 across various benchmarks.

2 Related Work

Reinforcement Learning for LLMs. Ouyang et al. (2022) trains a reward model using preference data and employs Proximal Policy Optimization (PPO) (Schulman et al., 2017) to perform reinforcement learning on LLMs for alignment with human preferences. Subsequently, many new approaches have employed reinforcement learning to align LLMs with human preferences (Wang et al., 2024). DeepSeekMath (Shao et al., 2024) proposed the GRPO algorithm, which simplifies the training process of reinforcement learning and significantly enhances the performance of LLMs in the mathematical domain through RL. Subsequently, DeepSeek-R1 (Guo et al., 2025) and Kimi k1.5 (Team et al., 2025) successfully demonstrated the substantial impact of reinforcement learning combined with rule-based reward sets in enhancing the reasoning capabilities of models. Liu et al. (2025) and Yu et al. (2025), among others, further introduced improvements to optimization algorithms to enhance training effectiveness.

Exploration and Exploitation in Reinforcement Learning. Balancing exploration and exploitation is a central challenge in reinforcement learning. Common strategies include ε-greedy, Upper Confidence Bounds (UCB), and Boltzmann Exploration (Sutton et al., 1998). Boltzmann Exploration selects actions based on a softmax probability distribution proportional to the exponential of the estimated values of actions, regulated by a temperature parameter τ. Similarly, LLMs generate tokens using a softmax distribution. Asadi and Littman (2017) introduced the Mellowmax operator to enhance the stability of Softmax, while Kim and Konidaris (2019) applied meta-gradient reinforcement learning to dynamically adjust temperature parameter of Mellowmax for better exploration-exploitation trade-offs. Moreover, Entropy Regularization, a common technique in deep reinforcement learning, adds an entropy term to the optimization objective to encourage stochastic policies and broader exploration (Williams, 1992). Apart from these reinforcement learning-based methods, a self-improvement approach Zeng et al. (2024) similarly enhances training performance by maintaining the exploration capability of the LLM through adjustments among several discrete temperature levels.

3 Preliminary

3.1 Group Relative Policy Optimization (GRPO)

We utilize the GRPO (Shao et al., 2024) algorithm to optimize the policy π_θ (LLMs). GRPO estimates the advantage in a group-relative manner. Specifically, given a question-answer pair $(q, a) \sim \mathcal{D}$, the old policy $\pi_{\theta_{\text{old}}}$ generates G individual responses $\{o_i\}_{i=1}^{G}$ and then optimizes the policy model by maximizing the following objective:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) =& \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\ & \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\right. \\ & \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\Big) \\ & \left.-\beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Big)\right], \qquad (1)\end{aligned}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_i,<_t)}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_i,<_t)}$ and the advantage $\hat{A}_{i,t}$ is computed as:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}. \quad (2)$$

Here, r_i is the reward of response o_i. The term $D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ represents the KL divergence penalty, which is used to prevent the policy from deviating excessively from the initial policy π_{ref}. Following prior work (Yu et al., 2025; Liu et al., 2025), we do not employ this penalty term. Therefore, we do not elaborate further on this detail.

4 Methodology

In this section, we first introduce our method for modeling question difficulty and propose a dynamic rollout budget allocation mechanism to allocate budgets based on question difficulty, improving training efficiency and the model's ability to answer complex questions. Next, we introduce a temperature scheduler to maintain the policy entropy, enhancing exploration, and further combine it with an annealing mechanism to balance exploration and exploitation.

4.1 Dynamic Rollout Budget Allocation

More challenging questions require a greater number of samples to obtain the correct answer. To allocate computational resources more efficiently, we transfer the rollout budget from simpler questions to more difficult ones, thereby enhancing the model's ability to address challenging problems.

We first introduce the method for modeling question difficulty. The RL training dataset $\mathcal{D} = \{(q_1, a_1), (q_2, a_2), \ldots, (q_{|\mathcal{D}|}, a_{|\mathcal{D}|})\}$ consists of questions q_i, their corresponding ground truth answers a_i. For each question q_i, its cumulative rollout count n_i^c and cumulative reward r_i^c are recorded. At the end of each dataset iteration, data points are ranked by their average reward $\frac{r_i^c}{n_i^c}$. The descending order of q_i's average reward is $\text{rank}(q_i)$, and its normalized ranking is $k_i = \frac{\text{rank}(q_i)}{|\mathcal{D}|}$, where a larger k_i indicates higher difficulty.

After defining the difficulty of the questions, we allocate the rollout budget based on the identified difficulty levels. Specifically, we define the default, minimum, and maximum sampling budgets as G, G_{min}, and G_{max}, respectively. The sampling budget G_i for question q_i is determined based on k_i, such that larger k_i values correspond to higher allocated rollout budgets. The dynamic sampling budget allocation process is detailed in Algorithm 1, which ensures that the total rollout budget within a batch remains constant.

Algorithm 1 Dynamic Rollout Budget

Require: A batch of rankings $\{k_{(1)}, \ldots, k_{(B)}\}$, G, G_{max}, G_{min}

1: Total rollout budget $N_{\text{total}} = B \times G$
2: For i in $\{1, 2, \ldots, B\}$, initialize $G_{(i)} = G_{\text{min}}$
3: Remaining rollouts budget $N_{\text{rem}} = N_{\text{total}} - B \times G_{\text{min}}$
4: For i in $\{1, 2, \ldots, B\}$, $G_{(i)} = G_{(i)} + \lfloor N_{\text{rem}} \times \frac{k_{(i)}}{\sum_{j=1}^{B} k_{(j)}} \rfloor$
5: $N_{\text{rem}} = N_{\text{total}} - \sum_{i=1}^{B} G_{(i)}$
6: Distribute N_{rem} greedily based on descending order of k_i, respecting G_{max} for each $G_{(i)}$
7: **return** $\{G_{(1)}, \ldots, G_{(B)}\}$

To avoid inefficiencies in allocating higher budgets to challenging questions under an undertrained policy, G_{min} and G_{max} are initially set equal to G. After each iteration of $\mathcal{D}$, G_{max} is gradually increased, and G_{min} is progressively decreased until reaching predefined limits. This prevents overcommitting resources to difficult questions prematurely and is analogous to curriculum learning.

4.2 Temperature Scheduling to Promote Exploration

In reinforcement learning, policies may converge to local optima, hindering the discovery of the global optimum. By adding an entropy regularization term to the optimization objective, the strategy can be encouraged to explore more state and action (Mnih et al., 2016; Williams, 1992). The modified optimization objective is given by:

$$\mathcal{J}(\theta) = \mathcal{J}_{\text{GRPO}}(\theta) + \lambda \, \mathbb{E}_{(q,a)\sim\mathcal{D}, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} H(\pi_\theta(o_i|q)), \quad (3)$$

where $H(\pi_\theta(o_i|q))$ represents the Shannon entropy (Equation (6) for the specific definition) of the action (token) sampling distribution from the policy, and λ is the coefficient controlling the strength of the regularization term. Entropy measures the uncertainty of a distribution, providing an indication of the policy's level of exploration. However, rule-based rewards are very sparse. When the rewards of all rollouts for a question are identical, the advantage $\hat{A}_{i,t}$ is zero, resulting in the gradient of the GRPO optimization objective, $\nabla_\theta J_{\text{GRPO}}(\theta)$,

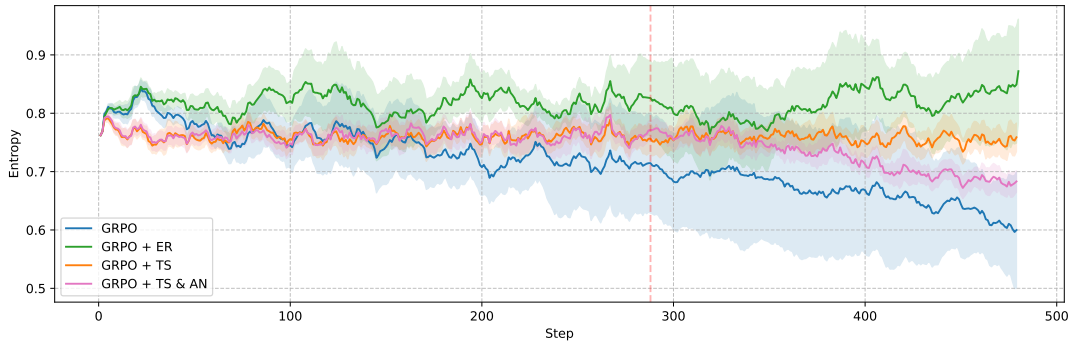

Figure 1: Smoothed entropy variations during training under different configurations. The curves represent the mean values, while the shaded regions denote the standard deviation across multiple runs. Here, **ER** represents entropy regularization, **TS** refers to the temperature scheduler, and **AN** indicates annealing. The red vertical line indicates the step at which annealing begins.

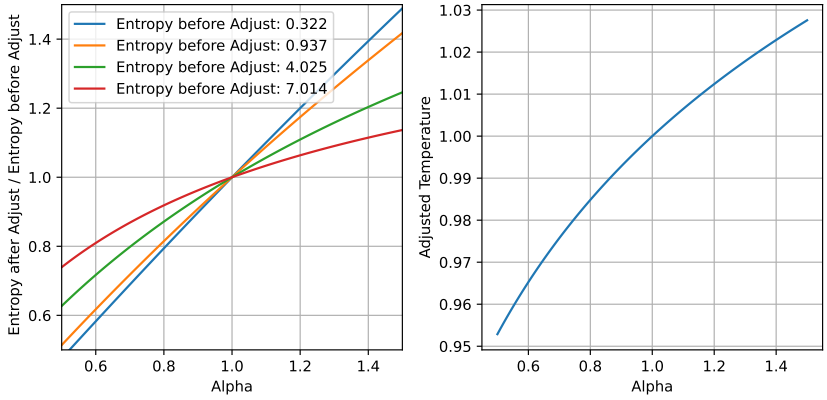

Figure 2: The left figure illustrates the relationship between the scaling factor of H_t, after temperature adjustment, and α. When the entropy is relatively small (the entropy magnitude of distribution for next token generation is typically on the order of 10^{-1}), the scaling factor closely approximates a linear relationship with α. The right figure illustrates the relationship between τ_{t+1} and α when $\tau_t = 1$.

is zero. In such cases, the update of the policy is primarily influenced by the gradient of the entropy regularization term, $\nabla_\theta H(\pi_\theta)$. As training progresses, cases where the advantage equals zero become more frequent (as the proportion of fully correct rollouts increases), at which point the gradient of entropy regularization may potentially lead to the gradual collapse of the policy.

Figure 1 shows the entropy evolution under different training setups. With GRPO alone, the policy's entropy declines rapidly, reducing the exploration of policy. The incorporation of entropy regularization effectively sustains higher entropy levels, thus fostering more diverse policy exploration. However, the experimental results in Section 5.3 show that the performance of the model trained with entropy regularization is even worse than that of the model trained with GRPO alone.

Temperature Scheduler. As discussed, although entropy regularization helps maintain the entropy of the policy, it may inadvertently introduce harmful gradients. To address this, we propose a temperature scheduler that adaptively adjusts the temperature τ of the softmax distribution to maintain policy entropy, ensuring stable exploration without introducing additional gradients. We aim to control the scaling of entropy by adjusting the temperature to maintain entropy at a stable level. However, the relationship between entropy and temperature is not linear. Fortunately, the entropy of the distributions for next token generation are typically small. Under this premise, we can adjust the temperature to precisely control entropy scaling using the following formula:

$$\tau_{t+1} = \tau_t \times \left(1 + \frac{\tau_t \ln \alpha}{\ln |\mathcal{V}| + \ln (\ln |\mathcal{V}|)}\right), \quad (4)$$

where $\alpha = \frac{H_{\text{init}}}{H_t}$, with H_{init} representing the aver-

age entropy of the first batch, which is the desired entropy level to maintain, and H_t denoting the average entropy at the current training step t. Additionally, $|\mathcal{V}|$ represents the vocabulary size of the LLM.

Formula (4) ensures that the scaling factor of entropy, after temperature adjustment, maintains an approximately linear relationship with α, as illustrated in Figure 2. **This indicates that entropy returns to the level of H_{init} after the temperature adjustment.** The detailed derivation of Formula (4) is provided in Appendix A. The temperature scheduler maintains the policy's entropy consistently at a stable level, as shown in Figure 1, thereby effectively enhancing policy exploration. Moreover, it is important to note that logits are also divided by the temperature during forward propagation after sampling, ensuring consistency between the LLM's distribution during training and sampling.

Annealing Mechanism. In the early stages of training, the policy requires sufficient exploration to avoid premature convergence to suboptimal solutions. As training progresses, we expect the policy to increasingly focus on exploiting high-value actions, thereby optimizing its performance more effectively. To balance exploration and exploitation, we introduce an annealing mechanism. Once the training step $t \geq t_{\text{anneal}}$, $\alpha = \frac{H^{(t)}_{\text{anneal}}}{H_t}$, where $H^{(t)}_{\text{anneal}}$ is calculated as follows:

$$H^{(t)}_{\text{anneal}} = H_{\text{init}} \cdot \left[\eta + (1-\eta) \cdot \frac{1}{2}\,(1+ \cos(\pi \cdot \frac{t - t_{\text{anneal}}}{t_{\max} - t_{\text{anneal}}}))\right]. \quad (5)$$

Here, $t_{\max}$ is the maximum training steps, $\eta \in [0, 1)$. Through this formula, We can gradually reduce the expect entropy level from H_{init} to $\eta \cdot H_{\text{init}}$. As shown in Figure 1, by introducing annealing, the entropy of the policy gradually decreases during the annealing phase. This facilitates a smooth transition of the policy from a high-entropy exploratory state to a lower-entropy exploitative state, ensuring that the policy maintains sufficient exploration during the early stages of training while gradually becoming more focused and efficient as training progresses.

5 Experiments

5.1 Setting

Training Datasets and Benchmarks. We follow DeepScaleR (Luo et al., 2025) in selecting MATH (Hendrycks et al., 2021), AIME 1983-2023 (of Problem Solving), Omni-MATH (Gao et al., 2024), and AMC (prior to 2023) as our training datasets. We follow Kimi K1.5 (Team et al., 2025) to enhance RL training efficiency by balancing the difficulty of the questions. In total, we collected 10k high-quality data points as the training set and 0.5k data points as the validation set. Further details are provided in Appendix B. We evaluate on AIME 2024, AMC 2023, MATH 500 (Hendrycks et al., 2021), and OlympiadBench (He et al., 2024).

Training Details. During sampling, the batch size is 64, with the default number of rollouts per question (G) set to 8. The sampling temperature is 1, and the maximum response length is 6k. Training is performed over 3 epochs on the 10k dataset, totaling 480 steps. We use DeepSeek-R1-Distill-Qwen 1.5B and 7B (Guo et al., 2025) as base models. For the 1.5B model, the learning rate is 5×10^{-6}, and for the 7B model, it is 2×10^{-6}. The policy update batch size is 64×8, and experiences from each sampling are used to update the policy only once. For the 7B model, training was conducted on 8 NVIDIA A100 GPUs, requiring approximately 8×36 GPU hours per experiment. For the 1.5B model, training was performed on 4 NVIDIA A100 GPUs, taking approximately 4×24 GPU hours per experiment. To ensure the reliability of results given the randomness in RL, each experiment was repeated 3 times. The training code is adapted from the VeRL framework (Sheng et al., 2024).

Evaluation Protocol. Unless otherwise specified, for each question, we default to sampling 16 times under the temperature of 1, with a maximum response length of 6k tokens. We use pass@1 and pass@16 (Chen et al., 2021) as our evaluation metric. **We report pass@16 because it reflects the model's potential to explore more solution paths to solve the questions.** The average metrics across the 3 runs are reported.

5.2 Main Results

In this section, we compare our approach, which integrates dynamic rollout budget allocation, temperature scheduling, and annealing, with the baselines.

Baselines. We select GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) as the baselines. Our proposed method is based on GRPO, which justifies its selection as a baseline. DAPO is a modified variant of GRPO, which filters out rollouts with zero advantage and obtains experience through multiple

Size	Method	AIME 2024		AMC 2023		MATH 500		Olympiad-Bench		Average	
		Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16
7B	GRPO	37.5	73.33	70.06	91.56	80.32	97.8	**53.72**	79.85	60.40	85.63
	DAPO	36.87	70.0	67.77	**95.18**	77.63	97.39	50.50	80.88	58.19	85.86
	Ours	**42.81**	**76.66**	**70.20**	93.57	**81.09**	**98.33**	53.70	**82.02**	**61.95**	**87.64**
1.5B	GRPO	24.66	59.76	60.73	88.79	**72.97**	95.86	46.33	74.66	51.17	79.76
	DAPO	19.16	60.0	52.86	84.33	68.81	94.6	41.17	70.81	45.50	77.43
	Ours	**27.70**	**66.66**	**62.95**	**90.36**	72.88	**96.39**	**46.35**	**75.40**	**52.47**	**82.20**

Table 1: Baseline comparison across different benchmarks.

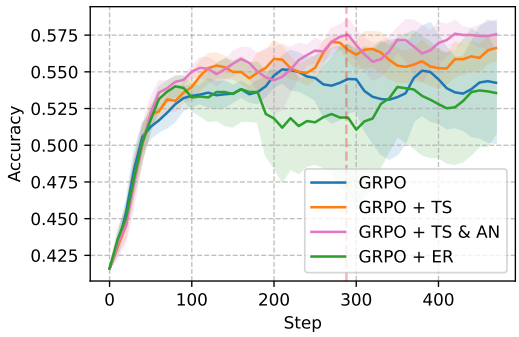

Figure 3: The accuracy on the validation set during training, with the shaded area representing the variance across multiple runs. The red vertical line indicates the step at which annealing begins.

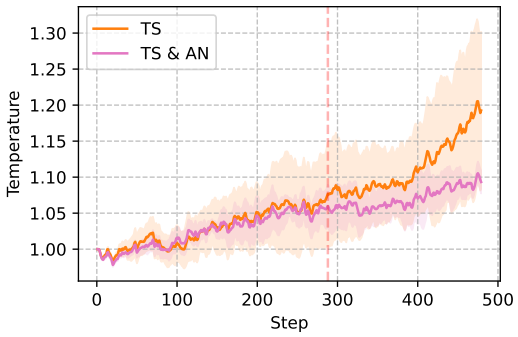

Figure 4: The temperature variation during training is presented for cases utilizing only the temperature scheduler and for those combining the scheduler with annealing.

rounds of online sampling. The general parameters for GRPO and DAPO are kept consistent with our method, while the DAPO-specific parameters are set to their default values as specified in its original paper.

Implementation Details. For dynamic rollout budget allocation (**DR**), $G_{\max}$ is increased by 2 and $G_{\min}$ is decreased by 2 after each epoch. For annealing (**AN**), we test $\eta \in \{0.8, 0.85, 0.9\}$, observing instability for $\eta = 0.8$ and $\eta = 0.85$. Consequently, η is set to 0.9, with the annealing start step at $\lfloor 0.6 \times t_{\max} \rfloor$.

Analysis. As shown in Table 1, for the 7B model, our method achieves advantages of 5.31% and 3.33% in pass@1 and pass@16, respectively, on the AIME benchmark. On other benchmarks, our method also demonstrates significantly higher pass@16 performance compared to GRPO, indicating that the models trained with our method possess greater exploratory potential. The models trained with our method also achieve the best average pass@1 and pass@16 across the four benchmarks. For the 1.5B model, our method exhibits similar advantages, demonstrating a significant improvement on the AIME benchmark and achieving the best average pass@1 and pass@16. The 1.5B model trained with DAPO performs significantly worse than those trained with other methods. This may be due to the 1.5B model's poor performance at the early stages of training, requiring numerous sampling rounds to gather enaugh experience for a single update. This results in substantial data being discarded, leading to its inferior performance. In contrast, GAPO and our method allow for more frequent policy updates, enabling faster improvement in model performance and more effective utilization of training data.

5.3 The Impact of the Temperature Scheduler and Annealing

In this section, we analyze the impact of the temperature scheduler (**TS**) on the training of reinforcement learning, and compare it to entropy regularization (**ER**). For entropy regularization, λ is set to 1×10^{-4}. For smaller benchmarks (AIME 2024 and AMC 2023), 128 answers per problem are sampled. For larger benchmarks (MATH 500 and Olympiad-Bench), the default sampling size of 16 is used.

Temperature Scheduler Maintains Entropy at a

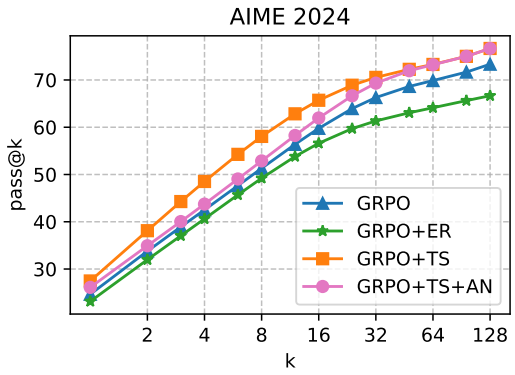

Figure 5: Pass@k on AIME 2024

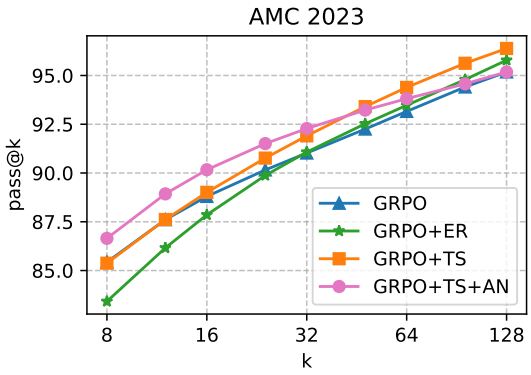

Figure 6: Pass@k on AMC 2023

Method	AIME 2024		AMC 2023		MATH 500		Olympiad-Bench		Average	
	pass@1	pass@16	pass@1	pass@16	pass@1	pass@16	pass@1	pass@16	pass@1	pass@16
GRPO	24.66	59.76	**60.73**	88.79	72.97	95.86	46.33	74.66	51.17	79.76
GRPO+ER	23.15	56.66	57.31	87.85	72.50	95.80	45.73	73.18	49.67	78.37
GRPO+TS	**27.15**	**65.55**	59.11	89.00	73.30	**96.73**	46.16	**76.04**	51.43	**81.83**
GRPO+TS+AN	26.11	61.95	59.91	**90.16**	**74.66**	96.20	**46.78**	74.41	**51.86**	80.68

Table 2: Comparison of different training methods on various benchmarks.

Stable Level. Figure 1 illustrates the entropy variation under different configurations during training. As discussed earlier, solely using GRPO results in a rapid entropy decline, leading to insufficient exploration. In contrast, training with a temperature scheduler maintains entropy at a stable level, facilitating greater exploration. The introduction of annealing gradually reduces the entropy of the policy during the annealing phase, enabling a transition from an exploratory state to a more efficient exploitation state. Figure 4 illustrates the temperature variation, where annealing slows the temperature increase during the annealing phase.

Temperature Scheduler Stabilizes LLM Performance Improvements. Figure 3 shows the variation in validation accuracy during training. Both GRPO alone and GRPO with entropy regularization exhibit significant variance. **In contrast, training with the temperature scheduler achieves lower variance and higher accuracy, indicating that temperature scheduling enhances training stability and effectiveness.** This is because increased exploration prevents the policy from becoming trapped in local optima, resulting in more stable performance improvements.

The Impact of the Temperature Scheduler on Performance. As shown in Figure 5 and Figure 6, models trained with the temperature scheduler generally outperform GRPO-only baselines in pass@k metrics, with the performance advantage consistently increasing as k grows. The experiments presented in Table 2 further demonstrate that incorporating the temperature scheduler significantly improves pass@16 compared to training solely with GRPO. In contrast, models trained with entropy regularization typically underperform relative to other methods, showing a slight advantage over GRPO-only training only when k is very large on the AMC benchmark.

What is the Role of Annealing? Figure 6 shows that annealing achieves greater improvements than temperature scheduling alone at lower k-values. However, as k increases, models with annealing are gradually surpassed by those using only temperature scheduling. A similar trend is observed in Table 2 on MATH 500 and Olympiad-Bench, where annealing outperforms at pass@1 but is overtaken by temperature scheduling alone at pass@16. On the more challenging AIME dataset, the temperature scheduler-only model consistently outperforms across all k-values, with the performance gap narrowing only at sufficiently high k. We attribute this phenomenon to this trade-offs: **Annealing improves precision on simpler questions by limiting the search space to high-value actions. In contrast, models trained exclusively with the**

Size	Method	AIME 2024		AMC 2023		MATH 500		Olympiad-Bench		Average	
		Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16	Pass@1	Pass@16
7B	All	**42.81**	**76.66**	70.20	**93.57**	81.09	**98.33**	**53.70**	**82.02**	**61.95**	**87.64**
	w/o DS	39.79	73.33	**70.48**	91.96	**81.15**	**98.33**	52.81	80.59	61.05	86.05
1.5B	All	**27.70**	**66.66**	**62.95**	**90.36**	72.88	**96.39**	46.35	**75.40**	**52.47**	**82.20**
	w/o DS	26.11	61.95	59.91	90.16	**74.66**	96.20	**46.78**	74.41	51.86	80.68

Table 3: Ablation Study Results on Dynamic Rollout Budget Allocation.

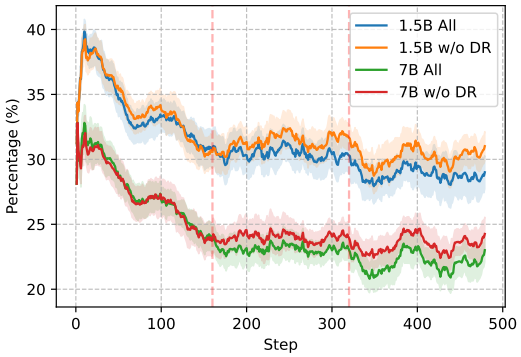

Figure 7: The variation in the proportion of questions for which all rollouts are incorrect (smoothed) during the training process. The red vertical lines indicate the intervals between different data iteration rounds, which also correspond to the points where G_{min} and G_{max} are adjusted.

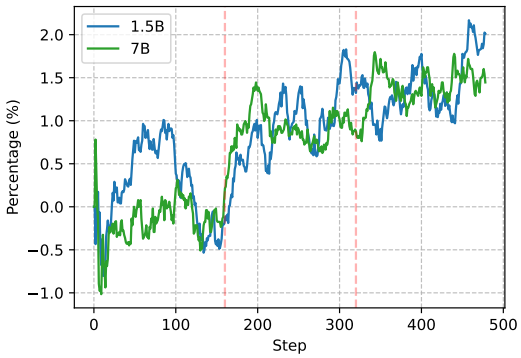

Figure 8: The difference in the proportion of questions for which all rollouts are incorrect (smoothed) between using dynamic rollout budgeting (DR) and not using DR during the training process.

temperature scheduler preserve a higher degree of exploratory capacity, facilitating the discovery of solution pathways for more complex questions.

5.4 Ablation Study on Dynamic Rollout Budget Allocation

In the experiments conducted in this section, we investigate the impact of dynamic rollout budget allocation.

The Impact of Dynamic Rollout Budget Allocation on Performance. As shown in Table 3, the performance of the model deteriorates significantly on the most challenging AIME benchmark when dynamic rollout budgeting is not employed. The pass@1 scores on the AIME benchmark decreased by 3.02% and 1.59% for the 7B and 1.5B models, respectively. On other benchmarks, performance generally declines when dynamic rollout budget allocation is not used, with only a few cases showing slight improvements.

The Impact of Dynamic Rollout Budget Allocation on the Proportion of Questions with All Incorrect Rollouts. Figure 7 illustrates the proportion of questions for which all rollouts are incorrect during the training process. During the first data iteration, dynamic rollout budget allocation is not yet activated, resulting in a similar proportion of entirely incorrect rollouts with or without dynamic budget allocation. During the second and third iterations, employing dynamic rollout budget allocation leads to a reduction in the proportion of questions for which all rollouts are incorrect. As shown in Figure 8, increase in G_{max} corresponds to a further reduction in the proportion of entirely incorrect rollouts. In the third iteration, when G_{max} is increased by 4 compared to G, a reduction of approximately 1.5% to 2% in the proportion of entirely incorrect rollouts is achieved.

6 Conclusion

In this paper, we propose a dynamic rollout budget allocation mechanism to enhance the efficiency of reinforcement learning and a temperature scheduler to encourage greater exploration by the model. We conduct experiments on models with 1.5B and 7B parameters. Experimental results demonstrate that our method significantly outperforms GRPO train-

ing on the most challenging AIME 2024 benchmark. Additionally, on other benchmarks, models trained with our method achieve substantially higher pass@16 scores compared to those trained with GRPO. This indicates that models trained using our approach retain exploratory capabilities, enabling them to uncover more potential correct paths.

Limitations

For the temperature scheduler, we have observed that annealing (low entropy) is more beneficial for simpler problems, while not using annealing (maintaining high entropy) is more advantageous for more challenging problems. Furthermore, we have modeled problem difficulty using the cumulative average reward. A natural idea, therefore, is to consider setting different temperatures for problems of varying difficulty. However, we have not conducted further experiments to explore this idea, leaving it as a direction for future work.

Due to computational resource constraints, we set the number of rollouts G per question to 8. However, increasing G and $G_{\max}$ could potentially amplify the effectiveness of dynamic rollout budget allocation.

Finally, although our approach can be easily extended to a broader range of reinforcement learning algorithms and domains, due to computational resource constraints, we limited our experiments to GRPO and the mathematics domain. Nevertheless, we believe that our method is sufficiently general and has the potential to be applied to other algorithms and domains.

References

Kavosh Asadi and Michael L Littman. 2017. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252. PMLR.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Chenghao Ma, Shanghaoran Quan, Liang Chen, Qingxiu Dong, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Seungchan Kim and George Konidaris. 2019. Adaptive temperature tuning for mellowmax in deep reinforcement learning. In *the NeurIPS 2019 Workshop on Deep Reinforcement Learning*.

Pranjal Kumar. 2024. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR.

Art of Problem Solving. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions. Accessed: 2025-05.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

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

Chenxia Tang, Jianchun Liu, Hongli Xu, and Liusheng Huang. 2024. Top- $n\sigma$: Not all logits are you need. *arXiv preprint arXiv:2411.07641*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, and 1 others. 2024. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *arXiv preprint arXiv:2407.16216*.

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

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Weihao Zeng, Yuzhen Huang, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. 2024. B-star: Monitoring and balancing exploration and exploitation in self-taught reasoners. *arXiv preprint arXiv:2412.17256*.

A Entropy Scaling through Temperature Adjustment in Softmax Distributions

In this section, we address the adjustment of the temperature τ such that the entropy is scaled by a factor of α. For simplicity, we focus on analyzing the distribution for the generation of a single next token. Let $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ represent the logits generated by a LLM, and define the temperature as $\tau > 0$. The commonly used Softmax probability distribution is defined as:

$$\mathbf{p} = [p_1, p_2, \ldots, p_N],$$

$$p_i = \frac{e^{z_i/\tau}}{\sum_{j=1}^{N} e^{z_j/\tau}}.$$

Let $z_{\max}$ be the maximum value in the logits, and define $\Delta_i = z_{\max} - z_i$. Then, p_i can be expressed as:

$$\begin{aligned} p_i &= \frac{e^{-(z_{\max}-z_i)/\tau}}{\sum_{j=1}^{N} e^{-(z_{\max}-z_j)/T}} \\ &= \frac{e^{-\Delta_i/T}}{\sum_{j=1}^{N} e^{-\Delta_j/\tau}} \\ &= \frac{e^{-\beta\Delta_i}}{Z(\beta)}, \end{aligned}$$

where $\beta = 1/\tau$ and $Z(\beta) = \sum_{j=1}^{N} e^{-\beta\Delta_j}$ .The Shannon entropy (Shannon, 1948) of this distribution is defined as:

$$H(\mathbf{p}) = -\sum_{i=1}^{N} p_i \ln p_i. \tag{6}$$

Substituting $\ln p_i = -\beta\Delta_i - \ln Z(\beta)$ into Equation (6) results in the following decomposition:

$$
\begin{aligned}
H(\mathbf{p}) =& -\sum_{i=1}^{N} p_i\big(-\beta\Delta_i - \ln Z(\beta)\big) \\
=& \beta \sum_{i=1}^{N} p_i\Delta_i \ + \ \ln Z(\beta) \sum_{i=1}^{N} p_i. \\
=& \beta \sum_{i=1}^{N} p_i\,\Delta_i \ + \ \ln Z(\beta).
\end{aligned}
$$

When the entropy $H(\mathbf{p})$ is small, the probability distribution $\mathbf{p}$ is primarily concentrated on a specific state, with the contributions from other states being negligible. Tang et al. (2024) analyzed the distribution pattern of logits from LLMs and observed that they typically consist of a Gaussian-distributed noisy region and a distinct informative region containing a few outlier tokens. For simplicity, the analysis focuses on the logits associated with the most informative token, specifically considering only z_{max}. z_{max} needs to be significantly larger than the logits in the noisy region to achieve a low entropy. We further assume that the difference between z_{max} and the logits in the noisy region is approximately equal, i.e., $\Delta_j \approx \Delta$ for $z_j < z_{\text{max}}$. Under this assumption, the normalization factor of the distribution can be expressed as:

$$
Z(\beta) \approx 1 + (N-1)e^{-\beta\Delta}.
$$

When the entropy is small, the probability corresponding to z_{max} is given by $p_{\text{max}} = \frac{1}{Z(\beta)} \approx 1$, which implies that $Z(\beta) \approx 1$. Consequently, for the remaining $N-1$ states, the probabilities can be approximated as:

$$
p_j = \frac{e^{-\beta\Delta}}{Z(\beta)} \approx e^{-\beta\Delta}.
$$

The entropy of the distribution can then be approximated as:

$$
\begin{aligned}
H(p) =& -\sum_{i}^{N} p_i \ln p_i \\
\approx& -(N-1)e^{-\beta\Delta}\ln(e^{-\beta\Delta}) \\
=& (N-1)\beta\Delta e^{-\beta\Delta}.
\end{aligned}
$$

Suppose the initial entropy is approximately represented as $\tilde{H}_0 = (N-1)\beta_0\,\Delta\,e^{-\beta_0\,\Delta}$. We aim to scale the entropy by adjusting the temperature τ, which is equivalent to modifying β. Suppose we scale this entropy by α times by adjusting β_0 to β_1, i.e.,

$$
\alpha\,(N-1)\beta_0\,\Delta\,e^{-\beta_0\,\Delta} = (N-1)\beta_1\,\Delta\,e^{-\beta_1\,\Delta}. \tag{7}
$$

Assuming $\beta_1\Delta = \beta_0\Delta + d$, the ratio becomes:

$$
\begin{aligned}
\alpha =& \frac{(N-1)\beta_1\Delta e^{-\beta_1\Delta}}{(N-1)\beta_0\Delta e^{-\beta_0\Delta}} \\
=& e^{-(\beta_1\Delta-\beta_0\Delta)}\,\frac{\beta_1\Delta}{\beta_0\Delta} \\
=& e^{d}\frac{\beta_0\Delta + d}{\beta_0\Delta}
\end{aligned} \tag{8}
$$

The change in entropy introduced by a single training step is typically minimal, meaning that the scaling factor α we aim to achieve is close to 1. we can further assume $d \approx 0$, allowing the approximation $\beta_0\Delta + d \approx \beta_0\Delta$, Equation (8) can be approximately expressed as: $\alpha \approx e^{-d}$. Then, by taking the natural logarithm, we obtain:

$$
d \approx -\ln\alpha. \tag{9}
$$

Thus, the ratio of the new temperature to the original temperature is:

$$
\begin{aligned}
\frac{\tau_1}{\tau_0} =& \frac{\beta_0}{\beta_1} = \frac{\beta_0\Delta}{\beta_1\Delta} \\
\approx& \frac{\beta_0\Delta}{\beta_0\Delta - \ln\alpha}.
\end{aligned} \tag{10}
$$

When the entropy is low, $\beta_0\Delta$ tends to be relatively large, while $\ln\alpha$ is close to zero. Thus, it follows that $\beta_0\Delta \gg \ln\alpha$. So, we can further approximate:

$$
\frac{\tau_1}{\tau_0} \approx \frac{\beta_0\Delta}{\beta_0\Delta - \ln\alpha} \tag{11}
$$

$$
\begin{aligned}
&= \frac{\beta_0\Delta - \ln\alpha + \ln\alpha}{\beta_0\Delta - \ln\alpha} = 1 + \frac{\ln\alpha}{\beta_0\,\Delta\, - \ln\alpha} \\
&\approx 1 + \frac{\ln\alpha}{\beta_0\Delta}.
\end{aligned} \tag{12}
$$

Therefore, the new temperature can be expressed as:

$$
\tau_1 \approx \tau_0 \times \left(1 + \frac{\ln\alpha}{\beta_0\Delta}\right) = \tau_0 \times \left(1 + \frac{\tau_0\ln\alpha}{\Delta}\right). \tag{13}
$$

Equation (13) describes the approximate relationship between temperatures before and after adjustment.

During training, logits can be utilized to estimate the value of Δ. However, computing Δ using logits incurs additional computational overhead and significant memory consumption, as the logits matrix is exceedingly large. To further simplify the computation, we introduce an approximation of the relationship between Δ and N, thereby eliminating the necessity of explicitly computing Δ during training. In this context, we disregard the effect of temperature, and the entropy is assumed to be a small value, ε. Based on the Equation (13), the entropy can be approximately expressed as:

$$\varepsilon \approx (N-1)\Delta e^{-\Delta}.$$

Taking the logarithm, we have:

$$\ln(\varepsilon) \approx \ln(N-1) + \ln(\Delta) - \Delta.$$

Rearranging terms, we obtain:

$$\Delta \approx \ln(N-1) + \ln(\Delta) - \ln(\varepsilon).$$

For sufficiently large N, $\ln(N-1)$ can be approximated as $\ln(N)$, leading to:

$$\Delta \approx \ln(N) + \ln(\Delta) - \ln(\varepsilon). \tag{14}$$

In Equation (14), the dominant term on the right-hand side is $\ln(N)$, while the other terms are comparatively smaller. Assuming $\Delta = \ln(N) + c$, we substitute this expression into Equation (14), yielding:

$$\ln(N)+c \approx \ln(N)+\ln(\ln(N)+c)-\ln(\varepsilon). \tag{15}$$

Simplifying Equation (15), we find:

$$c \approx \ln(\ln(N) + c) - \ln(\varepsilon).$$

For large N, the value of c is expected to be much smaller than $\ln(N)$. Thus, the addition of c to $\ln(N)$ does not significantly change the logarithm. For the distribution of the next token generated from LLMs, the magnitude of ε is typically on the order of 10^{-1}. Thus, we can further neglect the $\ln(\varepsilon)$ term, simplifying the expression. So c can be approximated as:

$$c \approx \ln(\ln(N)).$$

Therefore, $\Delta \approx \ln(N) + \ln(\ln(N))$. Substituting this approximation for Δ to Equation (13), the temperature scaling formula becomes:

$$\tau_1 \approx \tau_0 \times \left(1 + \frac{\tau_0 \ln \alpha}{\ln N + \ln(\ln N)}\right).$$

This provides an approximate formula for how the temperature needs to be adjusted to scale the entropy by a factor of α.

B Dataset details

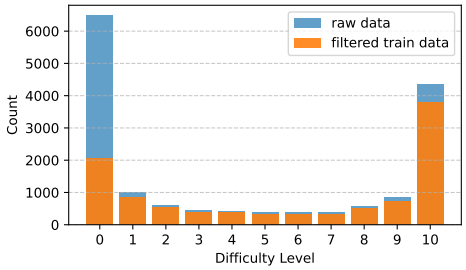

Figure 9: Difficulty distribution of the validation set. The orange color is the difficulty distribution of the filtered 10k data, and the blue color is the difficulty distribution of the original data.

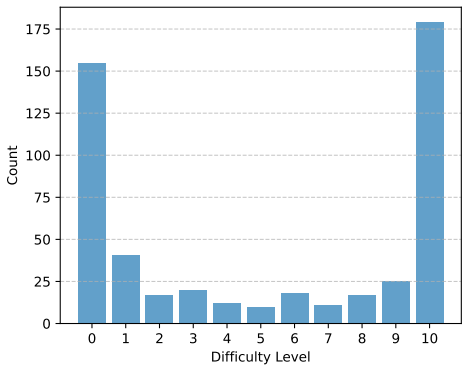

Figure 10: Difficulty distribution of validation set. The difficulty distribution of the validation set is consistent with the original data.

We further processed the DeepScaleR 40k dataset (Luo et al., 2025) to obtain our training data. Specifically, following the approach of Kimi k1.5 (Team et al., 2025), we improved the quality of reinforcement learning training data by reducing the proportion of simple questions.

We first utilized Qwen 2.5 Math 7B (Yang et al., 2024) to sample answers for each questions 10 times. The difficulty of the questions was assessed based on their accuracy rates. Subsequently, we balanced the data based on difficulty and filtered out a portion of simpler problems, capping the number of questions with 100% accuracy at 2k. This

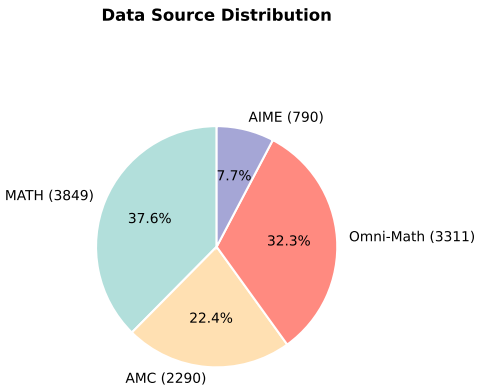

Figure 11: Pie chart of data sources.

balancing process not only ensures that the model is exposed to a diverse range of problem difficulties but also improves training efficiency by reducing the redundancy of overly simple problems, as training on simple problems is likely to yield minimal gains. The final 10k dataset exhibits a difficulty distribution as shown in Figure 9. The distribution of data sources is presented in Figure 11.

For the validation set, data was evenly sourced from the MATH, Omni-Math, AMC, and AIME datasets, with 128 samples from each dataset, resulting in a total of 512 samples. We did not balance the difficulty of the validation set, ensuring that its difficulty distribution closely resembles that of the original dataset, as shown in Figure 10. Furthermore, the validation set does not overlap with the training set.