MMR-GRPO: Accelerating GRPO-Style Training through Diversity-Aware Reward Reweighting

Kangda Wei, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University, College Station, TX
{kangda, huangrh}@tamu.edu

Abstract

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) has become a standard approach for training mathematical reasoning models; however, its reliance on multiple completions per prompt makes training computationally expensive. Although recent work has reduced the number of training steps required to reach peak performance (Yu et al., 2025), the overall wall-clock training time often remains unchanged or even increases due to higher per-step cost. We propose MMR-GRPO, which integrates Maximal Marginal Relevance to reweigh rewards based on completion diversity. The rationale is that redundant or similar completions, if repeatedly used to train a model, will create an "exploitation trap" and slow down model convergence in GRPO style reinforcement learning. Extensive evaluations across three model sizes (1.5B, 7B, 8B), three GRPO variants, and five mathematical reasoning benchmarks show that MMR-GRPO achieves comparable peak performance while requiring on average 47.9% fewer training steps and 70.2% less wall-clock time. These gains are consistent across models, methods, and benchmarks. Our code is released at: https://github.com/WeiKangda/MMR-GRPO.

1 Introduction

Recent advances in large language models (LLMs) have demonstrated remarkable capabilities in mathematical reasoning tasks (Wei et al., 2023; Wang et al., 2024), with reinforcement learning (RL) emerging as a critical technique for aligning models with complex reasoning objectives (Ahn et al., 2024; Wang et al., 2025; Trung et al., 2024; Kazemnejad et al., 2025; Setlur et al., 2024; Gehring et al., 2025; Li et al., 2024). Group Relative Policy Optimization (GRPO) or R1-Zero-style training (Shao et al., 2024; DeepSeek-AI et al., 2025) has become the de facto standard for training state-of-the-art mathematical reasoning models (Srivastava and Aggarwal, 2025), achieving strong performance across diverse benchmarks requiring multi-step problem-solving and logical inference. However, GRPO's reliance on generating multiple completions per prompt during training—typically 6-16 completions, or even 64 completions—makes it computationally expensive, requiring frequent model inference throughout the optimization process (Nimmaturi et al., 2025).

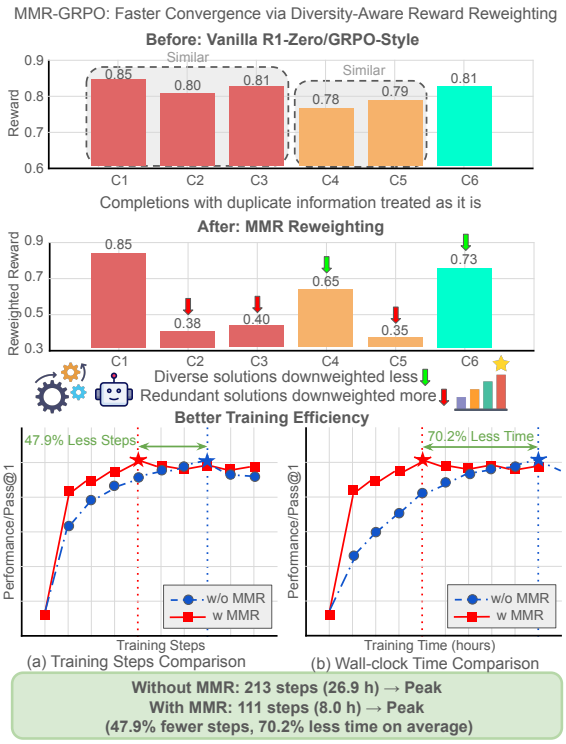

Figure 1: Before MMR, all completions receive similar rewards despite semantic redundancy (C1-C3 are similar, C4-C5 are similar). After MMR reweighting, diverse completions (C1, C4, C6) maintain high rewards while redundant ones (C2, C3, C5) are downweighted. Training with MMR achieves comparable peak performance with 47.9% fewer training steps and 70.2% less wall-clock time on average.

Training large reasoning models consumes substantial energy and computational resources, with carbon footprints growing proportionally to train-

Findings of the Association for Computational Linguistics: ACL 2026, pages 9584–9605
July 2-7, 2026

ing duration (Patterson et al., 2021; Faiz et al., 2024). Moreover, lengthy training time creates barriers for academic researchers and smaller organizations with limited GPU budgets, concentrating advanced reasoning capabilities among well-resourced institutions (Besiroglu et al., 2024; Ahmed and Wahed, 2020; Gelles et al., 2024). Methods that achieve comparable performance with reduced training time democratize access to state-of-the-art reasoning models while reducing environmental impact (Khandelwal et al., 2025).

In this work, we propose **MMR-GRPO**, a training-efficient approach that leverages Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to reweigh rewards based on completion quality and diversity, in order to promote exploration in reinforcement learning and accelerate model convergence. The rationale is that redundant or similar completions, if repeatedly used to train a model, will slow down model convergence in GRPO style reinforcement learning (RL). As a model is trained on its own generated completions in GRPO style RL, if a model keeps generating the same completion or highly overlapped completions, this shows that such completions align well with its current policy, then, training repeatedly with such highly redundant completions will create an "exploitation trap" and overly bias the model toward its current policy, limiting exploration of RL and slowing model convergence. Therefore, we strive to reduce redundancy within each group of completions to enable exploration in RL and achieve faster convergence. A similar effect is observed in deep RL, where diversity-based experience replay accelerates learning over redundant sampling (Zhao et al., 2025).

Specifically, inspired by the adoption of maximal marginal relevance in information retrieval for reducing redundancy among ranked documents, we propose **MMR-GRPO** to reduce redundancy among completions by balancing completion quality and diversity. As illustrated by Figure 1, MMR-GRPO reweighs rewards during advantage computation by penalizing completions that are semantically similar to higher-rewarded completions already being selected. More concretely, we compute sentence embeddings for all completions and apply a greedy MMR selection procedure that iteratively selects the next completion with the highest adjusted reward, adjusted based on its maximal semantic similarity with the already selected completions.

Further, we introduce a parameter-free adaptive mechanism where the diversity-quality trade-off λ is automatically determined by the standard deviation of completion rewards within each group: when rewards are tightly concentrated, diversity is prioritized to encourage exploration; when rewards are spread, quality is emphasized. This eliminates manual hyperparameter tuning while maintaining effectiveness across different model scales and training regimes.

We evaluate MMR-GRPO across three model sizes (1.5B, 7B, 8B parameters), three GRPO variants (the original GRPO, DR-GRPO and DAPO), and five mathematical reasoning benchmarks (MATH-500, AIME 2024, AMC 2023, Minerva Math, OlympiadBench). Our results demonstrate consistent and substantial training efficiency gains. MMR-GRPO achieves comparable performance as baseline methods while requiring 47.9% fewer training steps on average to reach peak performance. Despite MMR adding only 1-5% per-step computational overhead due to embedding computation and similarity matrix operations, the substantial reduction in total training steps translates to 70.2% wall-clock time savings overall.

In summary, our contributions include:

1. We propose **MMR-GRPO**, a diversity-aware reward reweighting approach that reduces both the number of training steps and wall-clock training time required to reach peak performance, while maintaining comparable performance. The method can be seamlessly integrated into GRPO–style training paradigms.
2. We introduce a parameter-free adaptive mechanism that balances rollout quality and diversity without hyperparameter tuning.
3. We provide comprehensive evaluation across three model scales, three methods, and five benchmarks, showing consistent efficiency gains.

2 Related Work

2.1 Reinforcement Learning for Mathematical Reasoning in LLMs

Reinforcement learning has emerged as a critical technique for aligning language models with complex reasoning objectives. Modern RL methods for LLMs evolved from foundational algorithms—Vanilla Policy Gradient (Williams, 2004), TRPO (Schulman et al., 2017a), and PPO (Schulman et al., 2017b)—into specialized variants for

language model training, such as GRPO (Shao et al., 2024) and later DAPO (Yu et al., 2025). For mathematical reasoning specifically, GRPO (Shao et al., 2024), or R1-Zero-style training (DeepSeek-AI et al., 2025), has become the de facto standard. Recent extensions of GRPO prioritizes algorithmic improvements for better performance (Li et al., 2025a; Chen et al., 2025b; Nan et al., 2025; Bamba et al., 2025) over training efficiency—the focus of our approach.

DAPO (Yu et al., 2025) also proposes dynamic sampling, a technique that repeatedly generates candidate samples and discards low-variance sample sets until a high-variance sample set is obtained. Dynamic sampling was proposed to improve training efficiency and reduce the total number of training steps needed for model convergence. However, even though dynamic sampling reduces training steps by 50-67%, recent analysis reveals a critical limitation: each dynamic sampling step requires approximately 1.5~3 times longer wall-clock time (Li et al., 2025a; Lian, 2025) due to multiple rounds of generation and filtering (NVIDIA, 2025). Consequently, even though dynamic sampling reduces training steps, DAPO's actual wall-clock training time remains comparable to or even exceeds vanilla GRPO (Chen et al., 2025a), limiting its practical efficiency gains.

2.2 Low-Variance Sample Groups in GRPO

In GRPO-style training, each update is based on a group of sampled completions, with policy gradients computed from the relative rewards within the group. However, when the sampled group exhibits low reward variance—for instance, when all completions are similarly incorrect or similarly correct—the resulting advantages become weak or uninformative, leading to inefficient policy updates and slower convergence. Prior work has addressed this issue through special handling or avoidance of low-variance groups. DAPO (Yu et al., 2025) and Xrpo (Bamba et al., 2025) simply discard low-variance sample sets, reducing the number of training steps required to reach peak performance. CPPO (Lin et al., 2025) accelerates GRPO by pruning completions with low absolute advantages to reduce forward passes during policy updates, though it operates on sample selection and does not modify how retained completions are rewarded, which is orthogonal to our approach. Other methods modify the learning signal to better exploit low-variance or negative samples, by separating the loss computation for positive and negative samples (Zhu et al., 2025), or introducing a virtual perfect-score reward to reshape advantages when all rewards are zero (Nan et al., 2025).

In contrast, our method directly exploits low-variance groups rather than discarding or treating them differently. By uniformly reweighing samples based on semantic diversity for any group of samples, we extract informative learning signals even when the reward variance of a group is low. As a result, our approach accelerates learning and not only reduces the number of training steps required for convergence, but also achieves consistent wall-clock time savings.

2.3 Diversity-Aware Methods for GRPO

Several concurrent works also incorporate diversity signals into GRPO training. DIVER (Hu et al., 2026) introduces an intrinsic diversity reward via potential-based reward shaping, but requires maintaining a potential function across steps and heuristics to mitigate reward hacking. DARLING (Li et al., 2025b) trains a semantic equivalence classifier to cluster completions for diversity scoring, requiring domain-specific annotations.

DRA-GRPO (Chen et al., 2025b) reweighs the rewards of all the completions using Submodular Mutual Information (SMI) solely based on pairwise cosine similarities or levels of redundancy. MMR-GRPO shares DRA-GRPO's lightweight design but differs in the reweighting mechanism by considering both completion redundancy and completion quality. After reward reweighting of DRA-GRPO, there may not be any high quality completion retaining a high reward if the high quality completions all have high similairities with each other or the other completions in the group, on the other hand, a low quality completion may gain a relatively high reward just because the completion is unique. In contrast, MMR-GRPO reduces redundancy in a group of completions by balancing completion quality with completion diversity. In our comparison experiments (Section 5.2) adopting the robust multi-sampling approach ($n > 1$) for calculating the metric pass@1, our approach MMR-GRPO consistently outperforms DRA-GRPO in maintaining the peak performance and in significantly reducing training steps.

3 Method

3.1 Background: GRPO

GRPO (Shao et al., 2024) is an effective reinforcement learning method for aligning language models with human preferences. Unlike traditional policy gradient methods, GRPO optimizes the policy by generating multiple responses and computing advantages within a group of responses.

Given a prompt x, GRPO generates G completions $\{y_1, y_2, \ldots, y_G\}$ and computes a reward $r(y_i)$ for each completion using a reward model. The advantage for each completion is calculated as:

$$A(y_i) = \frac{r(y_i) - \mu_G}{\sigma_G + \epsilon} \tag{1}$$

where μ_G and σ_G are the mean and standard deviation of rewards within the group, and ϵ is a small constant for numerical stability.

The training objective combines the policy gradient with a KL divergence penalty to prevent the policy from deviating too far from a reference model:

$$\mathcal{L} = -E_{y\sim\pi_\theta}\left[\log \pi_\theta(y|x) \cdot A(y) - \beta \cdot D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right] \tag{2}$$

where π_θ is the trainable policy, π_{ref} is the reference policy, and β is the KL penalty coefficient.

While GRPO effectively leverages group-relative rewards, it does not explicitly account for diversity among the generated completions. High-reward responses that are semantically similar may dominate the training signal, leading to slower convergence.

3.2 MMR-based Reward Reweighting with λ

To address the lack of diversity consideration in vanilla GRPO, we propose a MMR inspired reward reweighting mechanism. MMR (Carbonell and Goldstein, 1998) was originally designed for document retrieval to balance relevance and diversity. We adapt this principle to reward shaping in GRPO. The complete algorithm is shown in Algorithm 1.

Let $\mathcal{E}(y_i) \in R^d$ denote the embedding of completion y_i, obtained using a pre-trained sentence encoder. We define the cosine similarity between two completions as $m(y_i, y_j) = \mathcal{E}(y_i)^\top \mathcal{E}(y_j)$.

The MMR-based reward adjustment follows a greedy selection procedure. We maintain a set $\mathcal{S}$ of selected completions and iteratively add completions that maximize a diversity-weighted score:

$$\text{score}(y_i) = \lambda \cdot r(y_i) - (1-\lambda) \cdot \max_{y_j \in \mathcal{S}} m(y_i, y_j) \tag{3}$$

where $\lambda \in [0, 1]$ is a hyperparameter controlling the trade-off between reward quality (relevance) and diversity, and is commonly set to 0.7. When $\lambda = 1$, the method reduces to standard reward-based selection; when $\lambda = 0$, it purely maximizes diversity.

The algorithm proceeds as follows:

1. Initialize $\mathcal{S} = \emptyset$ and precompute the similarity matrix $M \in R^{G\times G}$ where $M_{ij} = m(y_i, y_j)$.
2. Select the completion with the highest reward: $y^* = \arg\max_{y_i} r(y_i)$, and add it to $\mathcal{S}$.
3. For each remaining completion, compute its adjusted score based on Equation 3.
4. Select the completion with the highest score, add it to $\mathcal{S}$.
5. Repeat steps 3 and 4 until all completions are ranked.

Finally, return the adjusted scores as reweighted rewards $\tilde{r}(y_i)$, which will replace the original rewards $r(y_i)$ when computing advantages (Equation 1).

Note that after MMR reweighting, semantically similar correct answers will receive different advantages, which can be potentially confusing to the model. But we explain below that this is fine and preferred. The advantage assigned to a rollout will essentially become a multiplier on the token-level training loss, and this multiplier intuitively corresponds to the attention we would like the model to pay during training. For semantically similar correct answers, we prioritize learning from the first occurrence and subsequent repetitions should receive less attention to avoid over-training on an already familiar rollout, which could slow down exploration and learning. Meanwhile, low-quality answers will not receive a high advantage simply for being distinct as a low reward $r(y_i)$ will be assigned initially and the diversity term only modulates the original reward, according to Equation 3.

3.3 Parameter-free MMR Reweighting with Adaptive λ

While the λ-parameterized MMR reweighting is effective, it introduces an additional hyperparameter that requires tuning. To eliminate this dependency, we design an adaptive mechanism that automatically adjusts λ based on the distribution of rewards within each group.

The key insight is that when rewards have high variance, diversity is less critical because the model already explores different quality regions. Conversely, when rewards are similar, diversity becomes more important to avoid mode collapse. We

Algorithm 1 MMR-based Reward Reweighting

Input: Rewards $\{r(y_1), \ldots, r(y_G)\}$, L2-normalized embeddings $\{\mathcal{E}(y_1), \ldots, \mathcal{E}(y_G)\}$
Output: Reweighted rewards $\{\tilde{r}(y_1), \ldots, \tilde{r}(y_G)\}$

Compute adaptive λ: $\lambda_{\text{adapt}} \leftarrow \sigma(\text{std}(r)) = \frac{1}{1+e^{-\text{std}(r)}}$
Compute similarity matrix: $M_{ij} \leftarrow \mathcal{E}(y_i)^\top \mathcal{E}(y_j)$ for all $i, j \in [G]$ at once.
Initialize $\mathcal{S} \leftarrow \emptyset$
$i^* \leftarrow \arg\max_i r(y_i)$
Add i^* to $\mathcal{S}$ and set $\tilde{r}(y_{i^*}) \leftarrow r(y_{i^*})$
for $t = 1$ to $G - 1$ **do**
  for each $i \notin \mathcal{S}$ **do**
    $\text{best\_sim}_i \leftarrow \max_{j \in \mathcal{S}} M_{ij}$
  end for
  for each $i \notin \mathcal{S}$ **do**
    $\text{score}(y_i) \leftarrow \lambda_{\text{adapt}} \cdot r(y_i) - (1 - \lambda_{\text{adapt}}) \cdot \text{best\_sim}_i$
  end for
  $i^* \leftarrow \arg\max_{i \notin \mathcal{S}} \text{score}(y_i)$
  Add i^* to $\mathcal{S}$ and set $\tilde{r}(y_{i^*}) \leftarrow \text{score}(y_{i^*})$
end for
return $\{\tilde{r}(y_1), \ldots, \tilde{r}(y_G)\}$

formalize this intuition using the sigmoid function applied to the reward standard deviation:

$$\lambda_{\text{adapt}} = \sigma(\text{std}(r)) = \frac{1}{1+e^{-\text{std}(r)}} \tag{4}$$

where $\text{std}(r)$ denotes the standard deviation of rewards $\{r(y_1), \ldots, r(y_G)\}$ within a group.

This adaptive λ is **scale-invariant** as the sigmoid function automatically maps reward variability to a bounded range $(0.5, 1)$, making it robust to different reward scales across tasks. Empirically, when rewards are tightly concentrated (low $\text{std}(r)$), λ_{adapt} decreases toward 0.5, increasing the diversity penalty. When rewards are widely spread (high $\text{std}(r)$), λ_{adapt} approaches 1, prioritizing reward quality. This adaptive behavior provides a principled way to encourage diversity without sacrificing reward maximization when the model already exhibits sufficient exploration.

The parameter-free MMR reweighting follows the same greedy procedure as described in Section 3.2, but with λ replaced by λ_{adapt} computed automatically for each group of completions. This allows the model to dynamically balance relevance and diversity based on the reward landscape of each group. All the experiments in this paper are conducted with λ_{adapt}, unless noted otherwise.

4 Experimental Settings

4.1 Datasets

We evaluate our approach on five widely used mathematical reasoning benchmarks: AIME 2024, MATH 500 (Hendrycks et al., 2021), AMC 2023, Minerva (Lewkowycz et al., 2022), and Olympiad Bench (He et al., 2024). These benchmarks span competition-style and curriculum-aligned problems with varying levels of difficulty and reasoning complexity, providing a comprehensive evaluation to assess mathematical problem-solving and multi-step reasoning ability. Additional details about each benchmark are provided in Appendix A.1.

For training, we use the `knoveleng/open-rs` dataset (Dang and Ngo, 2025), which consists of mathematical reasoning problems paired with high-quality step-by-step solutions. The dataset covers a broad range of mathematical topics and difficulty levels and is well-suited for training models to generate coherent reasoning chains. Further details about the training data are described in Appendix A.2.

4.2 Evaluation Metrics

Following standard practice in evaluation from previous works (DeepSeek-AI et al., 2025; Yu et al., 2025; Li et al., 2025a), we adopt the **pass@k** metric to measure the probability that at least one correct solution exists among k generated candidates from n total samples. Formally, given $n = 16$ independently generated completions per problem, pass@k is computed as:

$$\text{pass@k} = E\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right] \tag{5}$$

where c is the number of correct solutions among the n samples. We primarily report **pass@1** with $n = 16$. When $k = 1$, this metric is equivalent to **avg@16**, which is the fraction of correct solutions among the 16 sampled completions.. Evaluations are done using *lighteval* [1] for reproducibility and fair comparison.

4.3 Models

We conduct experiments with three model scales from two different model families of Deepseek-AI[2]: DeepSeek-R1-Distill-Qwen-1.5B , DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B. All models are initialized from publicly available checkpoints from Huggingface. Larger models (7B and 8B) employ LoRA (Hu et al., 2021) for parameter-efficient fine-tuning (see Appendix B.1 for details). For embedding extraction, we use `jina-embeddings-v2-small-en`[3], which provide

[1] https://huggingface.co/docs/lighteval
[2] https://huggingface.co/deepseek-ai
[3] https://huggingface.co/jinaai/jina-embeddings-v2-small-en

strong semantic representations while maintaining computational efficiency.

4.4 Training Methods

We experiment with three GRPO-style reinforcement learning methods for aligning language models with mathematical reasoning objectives:

- **GRPO** (Shao et al., 2024) applies PPO-style policy gradient optimization (Schulman et al., 2017b) by generating multiple completions per prompt and computes advantages relative to group statistics, avoiding the need for a separate value network. We compare vanilla GRPO and MMR-GRPO, with both methods trained for 500 steps and evaluated every 50 steps.
- **DAPO** (Yu et al., 2025) introduces dynamic sampling, a technique that reduces training steps by discarding low-variance sample sets and regenerating samples during training. However, dynamic sampling incurs significant per-step computational overhead (Li et al., 2025a; Lian, 2025). To investigate whether MMR can serve as a more efficient alternative to dynamic sampling, we evaluate three DAPO configurations:
  - **DAPO**: Vanilla DAPO with dynamic sampling enabled, trained for 200 steps with evaluation after every 10 steps. The reduced training horizon is due to dynamic sampling's rapid convergence as well as its prohibitive per-step cost.
  - **DAPO-No-DS**: DAPO with dynamic sampling disabled, providing a controlled baseline without dynamic sampling. Trained for 500 steps with evaluation first after every 10 steps (0-200) and then after every 50 steps (200-500).
  - **MMR-DAPO-No-DS**: DAPO with MMR reweighting instead of dynamic sampling, using the same training and evaluation schedule as DAPO-No-DS. In this configuration, we do not discard any low-variance sample group, but we consistently reweigh rewards within each group of generated samples.

  This experimental design allows us to directly compare MMR and Dynamic Sampling as alternative training efficiency techniques while isolating their effects from DAPO's other algorithmic improvements.
- **DR-GRPO** (Liu et al., 2025) extends GRPO with de-biased optimization that reduce response-level length bias and question-level difficulty bias. We compare vanilla DR-GRPO and MMR-DR-GRPO, with both methods trained for 500 steps and evaluated every 50 steps.

Common training hyperparameters and other model details are provided in Appendix B.

5 Results and Discussions

5.1 MMR Impacts on Training Efficiency

Training Steps Reduction Table 1 demonstrates that MMR-based reward reweighting consistently reduces the number of training steps required to reach peak performance across all methods and model scales. For GRPO, MMR achieves comparable performance while reducing training steps from 350 to 150 steps for the 7B model (57% reduction) and from 350 to 50 steps for the 8B model (86% reduction), though the 1.5B model shows no reduction in training steps (both methods peak at 100 steps), which is mainly due to its smaller model size, as smaller models typically require fewer steps to converge. Similarly, for DR-GRPO, MMR-DR-GRPO reduces training steps from 150 to 100 steps (33% reduction) for 1.5B, from 300 to 50 steps (83% reduction) for 7B, and from 300 to 100 steps (67% reduction) for 8B models. For DAPO without dynamic sampling, MMR reduces steps from 200 to 170 (15% reduction) for 1.5B, from 200 to 100 (50% reduction) for 7B, and from 250 to 180 (28% reduction) for 8B models. On average, MMR achieves a 47.9% reduction in training steps while maintaining comparable peak performance. Figure 2 illustrates this faster convergence for 7B models: MMR variants reach their peak performance earlier than their baseline counterparts, and results for 8B and 1.5B models across all benchmarks (Figures 4 and 5) exhibit similar patterns (detailed in Appendix D.1).

Wall-clock Training Time Savings Beyond step reduction, MMR delivers substantial wall-clock training time savings, addressing a critical limitation of existing efficiency techniques. Table 1 reveals that DAPO with dynamic sampling, despite achieving peak performance in fewer nominal steps (90-160 steps), incurs drastically longer wall-clock times: 25.53 hours for 1.5B, 33.15 hours for 7B, and 93.75 hours for 8B models. This counterintuitive inefficiency stems from dynamic sampling's multi-round generation and filtering mechanism, which increases per-step computational cost by 5× compared to DAPO-No-DS, DAPO with no dynamic sampling. In contrast, MMR-DAPO-No-DS achieves similar peak performance while requiring only 7.72, 8.52, and 17.40 hours respectively—representing 70%, 74%, and 81% wall-

Size	Method/Model	AIME 24	MATH-500	AMC 23	Minerva	OlympiadBench	Average	Peak Step	Time (hrs)	Time per Step(s)
1.5B	DS-Distill-Qwen-1.5B	0.288	0.828	0.629	0.265	0.433	0.489	-	-	-
	GRPO	0.338	0.846	0.730	0.296	0.528	0.547	100	4.08	147
	MMR-GRPO	0.325	0.849	0.739	0.302	0.528	0.549	100	4.13	149
	DR-GRPO	0.335	0.844	0.744	0.297	0.523	0.549	150	6.11	147
	MMR-DR-GRPO	0.323	0.851	0.738	0.303	0.530	0.549	100	4.13	149
	DAPO	0.331	0.851	0.755	0.304	0.541	0.556	110	25.53	836
	DAPO-No-DS	0.348	0.855	0.744	0.298	0.529	0.555	200	8.93	161
	MMR-DAPO-No-DS	0.331	0.856	0.730	0.295	0.527	0.548	170	7.72	163
7B	DS-Distill-Qwen-7B	0.560	0.923	0.825	0.380	0.568	0.651	-	-	-
	GRPO	0.554	0.940	0.917	0.418	0.671	0.700	350	28.28	291
	MMR-GRPO	0.560	0.940	0.916	0.409	0.673	0.700	150	12.22	293
	DR-GRPO	0.565	0.939	0.914	0.420	0.672	0.702	300	24.30	292
	MMR-DR-GRPO	0.565	0.942	0.905	0.412	0.673	0.699	50	4.11	296
	DAPO	0.558	0.940	0.914	0.418	0.671	0.700	90	33.15	1326
	DAPO-No-DS	0.569	0.939	0.905	0.420	0.674	0.701	200	16.17	291
	MMR-DAPO-No-DS	0.567	0.941	0.920	0.417	0.672	0.703	100	8.52	307
8B	DS-Distill-Llama-8B	0.506	0.896	0.815	0.295	0.541	0.611	-	-	-
	GRPO	0.465	0.889	0.897	0.355	0.626	0.646	350	32.22	331
	MMR-GRPO	0.475	0.882	0.897	0.350	0.623	0.645	50	4.62	333
	DR-GRPO	0.488	0.895	0.881	0.351	0.632	0.649	300	27.72	333
	MMR-DR-GRPO	0.485	0.893	0.886	0.346	0.630	0.648	100	9.36	337
	DAPO	0.504	0.889	0.889	0.351	0.631	0.653	160	93.75	2109
	DAPO-No-DS	0.483	0.890	0.878	0.346	0.628	0.645	250	23.02	331
	MMR-DAPO-No-DS	0.477	0.888	0.878	0.353	0.631	0.646	180	17.40	348
Average Training Step Saved: 47.9%						**Average Training Time (hrs) Saved: 70.2%**				

Table 1: Peak performance (pass@1, $n = 16$) comparisons across model sizes (1.5B, 7B and 8B models) and training methods (GRPO, DR-GRPO and DAPO), before and after MMR reweighting. All metrics represent the best checkpoint for each configuration. The Peak Step indicates the training step where optimal average performance is achieved. Training time is logged based on wall-clock measurements on 2×NVIDIA H100 80GB GPUs. For MMR v.s. Non-MMR methods, blue = better but comparable; green = significantly better ($\geq$1% for performance, >25% for efficiency). Unshaded = equivalent. MMR reweighting consistently achieves comparable or better performance while requiring fewer training steps and less time, translating to substantial computational savings. DS is short for DeepSeek-R1.

clock time savings over vanilla DAPO. For GRPO and DR-GRPO, MMR similarly delivers dramatic time reductions: MMR-GRPO reduces training time from 28.28 to 12.22 hours (57% savings) for 7B and from 32.22 to 4.62 hours (86% savings) for 8B models, while MMR-DR-GRPO achieves even more impressive gains, reducing time from 24.30 to 4.11 hours (83% savings) for 7B and from 27.72 to 9.36 hours (66% savings) for 8B models. Across all configurations, MMR achieves an average of 70.2% wall-clock time savings—a substantial practical improvement that translates to lower computational costs, reduced carbon emissions, and improved accessibility for researchers with limited GPU budgets.

Minimal Per-step Computational Overhead A critical consideration for any training efficiency method is the per-step computational overhead it introduces. Table 1 shows that MMR adds minimal per-step cost: for 1.5B models, time per step increases from 147s to 149s for GRPO (+1.4%), from 147s to 149s for DR-GRPO (+1.4%), and from 161s to 163s for DAPO-No-DS (+1.2%). Similar patterns hold for 7B models and 8B models. This 1-5% overhead is negligible compared to the computational demands of model forward/backward passes and generation, making this modest per-step cost more than offset by the substantial reduction in total training steps. The contrast with DAPO's dynamic sampling is striking: while MMR adds 1-5% per step, dynamic sampling increases per-step time by 412% (836s vs 163s at 1.5B), 331% (1326s vs 307s at 7B), and 506% (2109s vs 348s at 8B), consistent with prvious works finding (Li et al., 2025a; Lian, 2025). This demonstrates that MMR achieves training efficiency gains through accelerated convergence rather than expensive per-step filtering.

5.2 Comparisons with DRA-GRPO

We further compare MMR-GRPO against DRA-GRPO (Chen et al., 2025b), another recent diversity-aware GRPO variant that reweighs completions based on semantic diversity. Table 2 reports peak avg@16 (pass@1, $n = 16$) and the corresponding peak training step for vanilla GRPO, DRA-GRPO, and MMR-GRPO across 1.5B, 7B,

Figure 2: Performance across training steps for 7B models across all three training methods (DAPO, GRPO, DR-GRPO). MMR variants consistently achieve faster convergence and reach peak performance with fewer training steps.

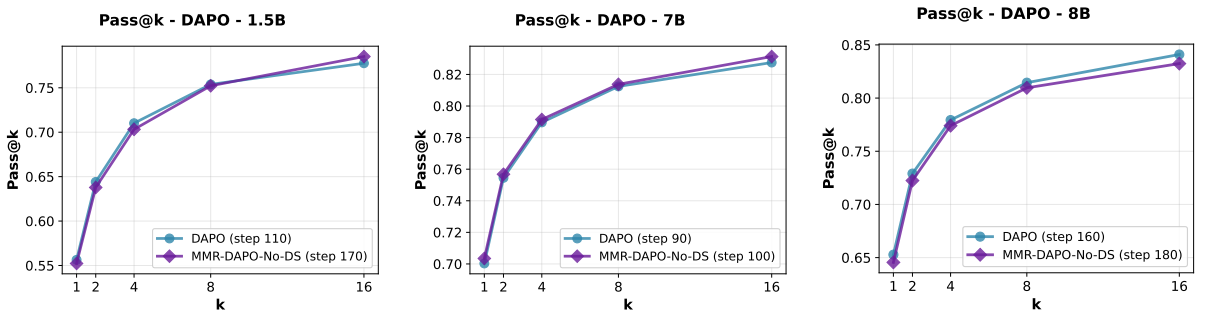

Figure 3: Pass@k curves for DAPO methods across three model scales (1.5B, 7B, 8B). MMR-DAPO-No-DS maintains nearly identical pass@k performance to vanilla DAPO across all k values (1, 2, 4, 8, 16), demonstrating that MMR training does not harm exploration capability.

and 8B models. Under the standard multi-sampling evaluation[4], DRA-GRPO consistently underperforms vanilla GRPO by a small amount. Regarding peak training steps, while DRA-GRPO has reduced the number of training steps on the 8B model, DRA-GRPO takes 100 more training steps on the 7B model to reach peak performance. In contrast, MMR-GRPO achieves comparable avg@16 and requires substantially fewer training steps, indicating that explicitly penalizing redundancy via MMR provides a stronger and more efficient training signal than DRA-GRPO.

Method	1.5B		7B		8B	
	Avg.	Step	Avg.	Step	Avg.	Step
GRPO	0.547	100	0.700	350	**0.646**	350
DRA-GRPO	0.545	100	0.699	450	0.644	150
MMR-GRPO	**0.549**	**100**	**0.700**	**150**	0.645	**50**

Table 2: Peak avg@16 (pass@1, $n = 16$) and peak training step for vanilla GRPO, DRA-GRPO, and MMR-GRPO across 1.5B, 7B, and 8B models.

5.3 MMR Effects on Trained Model Exploration Ability

A potential concern with using MMR is whether it might harm the model's ability to explore diverse solution strategies at inference time. Figure 3 ad-

[4]It appears to us that DRA-GRPO (Chen et al., 2025b) reported pass@1 results use k=1 and n = 1 (i.e., a single sample), which measures raw accuracy rather than the standard average@k metric with a larger sample size (e.g., k=16), which we and the broader community adopt.

Strategy	MMR-GRPO		MMR-DR-GRPO	
	Pass@1	Steps	Pass@1	Steps
Fixed λ=0.5	0.5495	150	0.5579	200
Fixed λ=0.6	0.5546	150	0.5515	50
Fixed λ=0.7	0.5548	200	**0.5605**	**150**
Fixed λ=0.8	0.5499	50	0.5470	50
Fixed λ=0.9	**0.5636**	**100**	0.5497	50
Adaptive λ	**0.5583**	**100**	**0.5589**	**100**

Table 3: Ablation study on the diversity-relevance trade-off parameter λ for 1.5B models. Pass@1 scores are computed with n=1 due to budget restriction.

dresses this concern by comparing pass@k curves (k=1 to 16) for DAPO and MMR-DAPO-No-DS across all three model scales. The curves are nearly overlapping across all k values, indicating that MMR-trained models maintain comparable exploration ability to their baseline counterparts. For instance, at k=16, all three methods achieve similar coverage (approximately 0.78 for 1.5B, 0.83 for 7B, and 0.83 for 8B models), suggesting that diversity-aware reward reweighting during training does not constrain the model's capacity to generate varied solutions during inference. This is intuitive: while MMR downweighs semantically redundant completions during training to accelerate learning, it does not restrict the model from learning novel reasoning paths. The trained model retains the ability to explore multiple reasoning paths, as evidenced by the comparable pass@k performance. Similar patterns hold for GRPO and DR-GRPO methods (Figures 6 and 7 in Appendix D.2). These results confirm that reweighting the completions rewards by using MMR to penalize completions with duplicate information does not restrict the model's exploration capabilities, making it a practical method for reducing training costs without sacrificing inference-time solution diversity.

5.4 Ablation Study

Adaptive λ Mechanism: Table 3 compares our adaptive λ mechanism against fixed values on 1.5B models. A key finding is that **the optimal fixed λ differs across methods**: λ=0.9 performs best for MMR-GRPO while λ=0.7 is optimal for MMR-DR-GRPO. This method-dependent sensitivity would require costly hyperparameter search for each new training algorithm. In contrast, adaptive λ, requiring no hyperparameter tuning, achieves competitive performance for both methods after a relatively small number of training step, further demonstrating robustness and the practical value of automatic adaptation.

	nomic	jina	Δ
AIME 2024	0.565	0.560	0.005
MATH-500	0.940	0.940	0.000
AMC 2023	0.914	0.916	-0.002
Minerva	0.412	0.409	0.003
OlympiadBench	0.673	0.673	0.000
Average	0.701	0.700	0.001
Peak Step	150	150	0
Time (hrs)	12.30	12.22	0.08

Table 4: Embedding model robustness ablation on MMR-GRPO. nomic = nomic-embed-text-v1.5; jina = jina-embeddings-v2-small-en.

Robustness to Embedding Model: A natural question is whether our method is sensitive to the choice of embedding model used for computing semantic similarity. To investigate this, we compare our default embedding model, `jina-embeddings-v2-small-en` against `nomic-embed-text-v1.5` on MMR-GRPO using 7B model, keeping all other hyperparameters identical. As shown in Table 4, the two models yield nearly indistinguishable results: the average benchmark score differs by only 0.1%, peak performance is reached at the same training step, and wall-clock time differs by less than 1%. Per-benchmark scores are equally consistent, with a maximum deviation of 0.5% on AIME 2024. These results confirm that our method is robust to the choice of embedding model, as the MMR reweighting mechanism relies on *relative* similarity rankings among completions rather than absolute embedding magnitudes, making it insensitive to embedding model-specific representation differences.

6 Conclusions

We propose MMR-GRPO, a diversity-aware reward reweighting method for accelerating model learning in GRPO-style training. By downweighting semantically redundant completions using Maximal Marginal Relevance, MMR-GRPO provides more informative policy updates and accelerates convergence. Across three model sizes, three GRPO variants, and five benchmarks, MMR-GRPO matches peak performance while reducing training steps by 47.9% and wall-clock time by 70.2%. Overall, MMR-GRPO lowers training cost while preserving performance, democratizing access to advanced reasoning model development and reduces the environmental impact of RL training.

Limitations

While our proposed MMR-based reward reweighting achieves comparable performance to baseline methods with significantly fewer training steps and less training time, we acknowledge several limitations.

First, the greedy MMR selection algorithm has quadratic time complexity $\mathcal{O}(N^2)$, where N is the number of generated completions per prompt, compared to $\mathcal{O}(N)$ for standard GRPO reward normalization. However, this overhead is minimal in practice: our experiments use $N = 6$ generations per prompt (standard in prior works (Chen et al., 2025b; Dang and Ngo, 2025)), the MMR reweighting adds only 1 - 5% to training time per step since model generation and forward passes remain the dominant computational costs. Our fully vectorized GPU implementation leverages efficient batched matrix multiplication, even with $N = 64$ ($64^2 \times 512 = 2.1$M FLOPs for $N = 64$ with 512-dimensional embeddings), making the similarity matrix computation negligible compared to a single forward pass through a multi-billion parameter language model. For context, a single forward pass through a 1.5B parameter model requires approximately 3×10^9 FLOPs (assuming 2 FLOPs per parameter). The MMR computation at N=64 represents only 0.07% of a single forward pass. Moreover, the similarity matrix is computed once per training step using fully vectorized GPU operations (batched matrix multiplication), adding negligible wall-clock overhead. Even scaling to N=128 or N=256 would remain practically insignificant compared to the model's generation and gradient computation costs. Importantly, the computational overhead of MMR is more than offset by the reduced training steps required to reach peak performance: our method achieves optimal results in 47.9% fewer steps on average, resulting in substantial wall-clock time savings despite the per-step overhead.

Second, due to computational constraints, our experiments are limited to models up to 8B parameters using LoRA fine-tuning for larger scales. We have not evaluated our approach on larger models (14B, 32B, or 70B parameters) with full fine-tuning. Given that our method's efficiency gains stem from encouraging diverse exploration early in training, we hypothesize that larger models with greater capacity for memorizing diverse solution strategies may benefit even more from MMR reweighting, though empirical validation remains future work.

Finally, our evaluation focuses on mathematical reasoning benchmarks; the generalization of MMR-based reward reweighting to other domains such as code generation or commonsense reasoning, which may exhibit different convergence dynamics and solution space characteristics, remains to be explored.

Ethics and Broder Impacts

This work improves the training efficiency of reinforcement learning for mathematical reasoning models, achieving comparable performance with 47.9% fewer training steps and 70.2% less training time. This reduction in computational requirements has positive environmental implications by lowering energy consumption and carbon emissions, and democratizes access to advanced reasoning model development for researchers with limited budgets. However, we acknowledge that more efficient training of capable reasoning models could accelerate deployment in high-stakes domains such as educational assessment and automated theorem proving, which require careful validation and human oversight to prevent misuse (e.g., automated exam cheating). Our evaluation is limited to English-language mathematical benchmarks; equitable access to reasoning AI requires extending such methods to multilingual settings. We encourage practitioners to implement appropriate safeguards and emphasize that responsible AI development requires investments in robustness testing, bias mitigation, and value alignment beyond computational efficiency.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback and input. We gratefully acknowledge support from National Science Foundation via the award IIS-1942918. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High-Performance Research Computing.

References

Nur Ahmed and Muntasir Wahed. 2020. The democratization of ai: Deep learning and the compute divide in artificial intelligence research. Preprint, arXiv:2010.15581.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Pro-

gresses and challenges. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Udbhav Bamba, Minghao Fang, Yifan Yu, Haizhong Zheng, and Fan Lai. 2025. Xrpo: Pushing the limits of grpo with targeted exploration and exploitation. Preprint, arXiv:2510.06672.

Tamay Besiroglu, Sage Andrus Bergerson, Amelia Michael, Lennart Heim, Xueyun Luo, and Neil Thompson. 2024. The compute divide in machine learning: A threat to academic contribution and scrutiny? Preprint, arXiv:2401.02452.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Weizhe Chen, Sven Koenig, and Bistra Dilkina. 2025a. Lspo: Length-aware dynamic sampling for policy optimization in llm reasoning. Preprint, arXiv:2510.01459.

Xiwen Chen, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Hao Wang, Haiyu Wu, Huayu Li, Aristeidis Sotiras, Yalin Wang, and Abolfazl Razi. 2025b. Dra-grpo: Exploring diversity-aware reward adjustment for r1-zero-like training of large language models. Preprint, arXiv:2505.09655.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. Preprint, arXiv:2503.16219.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Preprint, arXiv:2501.12948.

Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. 2024. Llmcarbon: Modeling the end-to-end carbon footprint of large language models. Preprint, arXiv:2309.14393.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. 2025. Rlef: Grounding code llms in execution feedback with reinforcement learning. Preprint, arXiv:2410.02089.

Rebecca Gelles, Veronica Kinoshita, Micah Musser, and James Dunham. 2024. Resource democratization: Is compute the binding constraint on ai research? Proceedings of the AAAI Conference on Artificial Intelligence, 38(18):19840–19848.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

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. Preprint, arXiv:2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. Preprint, arXiv:2106.09685.

Zican Hu, Shilin Zhang, Yafu Li, Jianhao Yan, Xuyang Hu, Leyang Cui, Xiaoye Qu, Chunlin Chen, Yu Cheng, and Zhi Wang. 2026. Diversity-incentivized exploration for versatile reasoning. Preprint, arXiv:2509.26209.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2025. Vineppo: Refining credit assignment in rl training of llms. Preprint, arXiv:2410.01679.

Apoorv Khandelwal, Tian Yun, Nihal V. Nayak, Jack Merullo, Stephen H. Bach, Chen Sun, and Ellie Pavlick. 2025. $100k or 100 days: Trade-offs when pre-training with academic resources. Preprint, arXiv:2410.23261.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. Preprint, arXiv:2309.06180.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. Preprint, arXiv:2206.14858.

Gang Li, Ming Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. 2025a. Disco: Reinforcing large reasoning models with discriminative constrained optimization. Preprint, arXiv:2505.12366.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. 2025b. Jointly reinforcing diversity and quality in language model generations. Preprint, arXiv:2509.02534.

Ziniu Li, Tian Xu, and Yang Yu. 2024. Policy optimization in RLHF: The impact of out-of-preference data. In The Second Tiny Papers Track at ICLR 2024.

Yongsheng Lian. 2025. Comparative analysis and parametric tuning of ppo, grpo, and dapo for llm reasoning enhancement. Preprint, arXiv:2512.07611.

ZhiHang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2025. CPPO: Accelerating the training of group relative policy optimization-based reasoning models. In The Thirty-ninth Annual Conference on Neural Information Processing Systems.

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. Preprint, arXiv:2503.20783.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and Xingzhong Xu. 2025. Ngrpo: Negative-enhanced group relative policy optimization. Preprint, arXiv:2509.18851.

Datta Nimmaturi, Vaishnavi Bhargava, Rajat Ghosh, Johnu George, and Debojyoti Dutta. 2025. Predictive scaling laws for efficient grpo training of large reasoning models. Preprint, arXiv:2507.18014.

NVIDIA. 2025. An In-depth Walkthrough of DAPO in NeMo RL. NVIDIA. Official NeMo RL documentation covering Dynamic Sampling implementation and its compute considerations.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. Preprint, arXiv:2104.10350.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. Preprint, arXiv:1502.05477.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. Preprint, arXiv:2410.08146.

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

Saksham Sahai Srivastava and Vaneet Aggarwal. 2025. A technical survey of reinforcement learning techniques for large language models. Preprint, arXiv:2507.04136.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Yi-Di Wang, Shu Yan, Cheng-Xing Jia, Xu-Hui Liu, Xin-Wei Chen, Jia-Cheng Xu, Ziniu Li, and Yang Yu. 2025. A survey on large language models for mathematical reasoning. Preprint, arXiv:2506.08446.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. Preprint, arXiv:2201.11903.

Ronald J. Williams. 2004. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8:229–256.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. Preprint, arXiv:2503.14476.

Kaiyan Zhao, Yiming Wang, Yuyang Chen, Yan Li, Leong Hou U, and Xiaoguang Niu. 2025. Efficient diversity-based experience replay for deep reinforcement learning. Preprint, arXiv:2410.20487.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in llm reasoning. Preprint, arXiv:2506.01347.

A Dataset Details

This section provides detailed information about the benchmarks used for evaluation and the training dataset.

A.1 Evaluation Benchmarks

We evaluate our approach on five mathematical reasoning benchmarks that cover diverse problem types, difficulty levels, and mathematical domains. Table 5 summarizes the key statistics of each benchmark.

Benchmark	# Problems	Difficulty
MATH-500	500	High School
AIME 2024	30	Competition
AMC 2023	40	High School Competition
Minerva Math	272	Undergraduate
OlympiadBench	675	Competition

Table 5: Statistics of evaluation benchmarks. Difficulty levels indicate the target audience or competition level.

MATH-500 (Hendrycks et al., 2021) A curated subset of 500 challenging problems from the MATH dataset, spanning seven mathematical domains: Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus.

AIME 2024 The American Invitational Mathematics Examination (AIME) 2024 consists of 30 problems designed for top-performing high school students who qualify through the AMC competitions. All answers are integers between 0 and 999, requiring precise numerical solutions. Problems demand multi-step reasoning, creative problem-solving strategies, and deep mathematical insight, making this one of the most challenging benchmarks in our evaluation suite.

AMC 2023 The American Mathematics Competitions (AMC) 2023 includes 40 problems from AMC 10 and AMC 12 contests. These problems cover fundamental to intermediate mathematical concepts including algebra, geometry, number theory, and combinatorics. Compared to AIME, AMC problems are more accessible but still require solid mathematical reasoning and problem-solving skills. Answers are typically integers or simple algebraic expressions.

Minerva Math (Lewkowycz et al., 2022) Originally curated to evaluate the Minerva model, this benchmark contains 272 undergraduate-level mathematics problems. Problems often involve complex multi-step derivations, symbolic manipulation, and application of advanced mathematical concepts.

OlympiadBench (He et al., 2024) A comprehensive collection of 675 competition-level mathematics problems sourced from international and national mathematical olympiads (IMO, USAMO, etc.). These problems represent the pinnacle of pre-collegiate mathematical problem-solving, often requiring creative insights, elegant proofs, and sophisticated mathematical techniques.

A.2 Training Dataset

We use the `knoveleng/open-rs` dataset (Dang and Ngo, 2025) for training all models. This dataset is specifically designed for reinforcement learning on mathematical reasoning tasks and contains high-quality step-by-step solutions following a structured format.

Dataset Composition The dataset consists of 1.7k mathematical problems paired with detailed reasoning chains. Each example includes:

- **Problem**: A natural language mathematical question
- **Reasoning process**: Step-by-step solution with the final answer in `\\boxed{}`
- **Final answer**: The value of the final answer

B Training Hyperparameters

This section provides detailed training configurations for all methods and model scales used in our experiments.

B.1 Model Configurations

Table 6 summarizes the architecture and parameter-efficient fine-tuning configurations for all three model scales.

B.2 Common Training Hyperparameters

Table 7 lists the training hyperparameters shared across all methods and model scales.

B.3 vLLM Generation Configuration

For efficient parallel generation during training, we use vLLM (Kwon et al., 2023) with the configuration shown in Table 8.

B.4 Reward Functions

Table 9 describes the reward functions used in our experiments.

B.5 Method-Specific Configurations

Table 10 compares the specific hyperparameters and configurations for each training method.

Configuration	1.5B	7B	8B
Model name	DeepSeek-R1-Distill-Qwen		DeepSeek-R1-Distill-Llama
Architecture	Qwen	Qwen	Llama
Parameters	1.5B	7B	8B
Fine-tuning method	Full FT	LoRA	LoRA
LoRA rank (r)	–	64	64
LoRA alpha (α)	–	128	128
LoRA dropout	–	0.05	0.05
LoRA target modules	–	q/k/v/o/gate/up/down proj	
Precision	bfloat16		
Attention	Flash Attention 2		

Table 6: Model architecture and PEFT configurations. All models are initialized from publicly available DeepSeek-R1 distilled checkpoints.

Hyperparameter	Value
Optimization	
Learning rate	1.0×10^{-6}
Optimizer	AdamW
LR scheduler	Cosine with min lr
Min LR ratio	0.1
Warmup ratio	0.1
Gradient clipping	1.0
Batch Configuration	
Batch size per device	6
Gradient accumulation steps	8
Effective batch size	48
Training Steps	
Max steps (GRPO/DR-GRPO/DAPO w/o DS)	500
Max steps (DAPO)	200
Logging steps	1
Sequence Lengths	
Max prompt length	512 tokens
Max completion length	3584 tokens
Max model length	4608 tokens
Generation	
Temperature	0.7
Number of generations (G)	6
System	
Random seed	2025
Gradient checkpointing	True
Mixed precision	bfloat16

Table 7: Common training hyperparameters across all methods and models.

B.6 System Prompt

All models use the following system prompt during training:

> *"A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer, and put your final answer within \boxed{} . The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. Note that respond by English, NOT use other languages."*

All models use the following system prompt during evaluation:

Parameter	Value
vLLM device	1 Auto-assigned GPU
GPU memory utilization	0.7
Max model length	4608 tokens
Eager mode	Enabled
Prefix caching	Enabled
Dtype	bfloat16

Table 8: vLLM configuration for efficient generation.

> *"Solve the following math problem efficiently and clearly. The last line of your response should be of the following format: 'Therefore, the final answer is: $\\boxed{{ANSWER}}$. I hope it is correct' (without quotes) where ANSWER is just the final number or expression that solves the problem. Think step by step before answering. "*

B.7 Computational Resources

All training experiments were conducted on 2xNVIDIA H100 80GB GPUs. All evaluation experiments were conducted on 1xNVIDIA A100 40GB GPU.

C Adaptive λ Design Choice

The design is motivated by two principles grounded in the MMR literature:

1. Lower bound of 0.5: The original MMR paper (Carbonell and Goldstein, 1998) establishes $\lambda = 0.7$ as the default, reflecting that relevance should generally dominate over diversity. Our sigmoid maps to $[0.5, 1)$, ensuring that even in the most diversity-favoring regime (low reward variance), reward quality always receives at least equal weight as diversity — quality is never subordinated.
2. Sigmoid function: The sigmoid naturally maps the unbounded reward standard deviation to a bounded range with smooth, monotonic behavior. When $std(r)$ is small (rewards

Reward Type	Description
Accuracy	Binary reward (1.0 or 0.0) based on whether the extracted answer matches the gold answer
Format	Measures compliance with the required output format (presence of <think> and <answer> tags, proper \boxed{} usage)
Cosine	This is an enhanced variant of the Accuracy Reward. It assigns rewards to model completions by jointly considering solution correctness and completion length, where the length-based scaling follows a cosine schedule. For each completion, both the model output and the reference solution are parsed. Correctness is determined by comparing the parsed representations. The final reward is computed as a cosine function of the completion length normalized by a predefined maximum length, which favors shorter correct responses while imposing stronger penalties on short but incorrect ones

Table 9: Description of reward functions used across different methods.

concentrated), λ approaches 0.5, increasing the diversity emphasis; when $std(r)$ is large (rewards spread), λ approaches 1, reverting to reward-dominated selection.

D Additional Results

D.1 Training Convergence for 8B and 1.5B Models

Figures 4 and 5 present the training convergence patterns for 8B and 1.5B models, respectively, complementing the 7B results shown in the main text (Figure 2). The convergence patterns observed in these models are consistent with those in the 7B model.

For 8B models (Figure 4), the efficiency gains are even more dramatic. MMR-GRPO achieves peak performance at step 50 compared to step 350 for vanilla GRPO (86% reduction), representing the largest step reduction observed across all configurations. MMR-DR-GRPO peaks at step 100 versus step 300 for vanilla DR-GRPO (67% reduction), while MMR-DAPO-No-DS requires 180 steps compared to 250 for DAPO-No-DS (28% reduction). These results suggest that the benefits of diversity-aware training may scale favorably with model capacity, as larger models can better exploit the informative signals from diverse completions.

For 1.5B models (Figure 5), MMR-GRPO reaches peak performance at step 100 compared to step 100 for vanilla GRPO, while MMR-DR-GRPO achieves peak performance at step 150 versus step 100 for vanilla DR-GRPO. The DAPO comparison shows that MMR-DAPO-No-DS peaks at step 170, representing a 15% reduction from the 200 steps required by DAPO-No-DS.

Across both model scales, the performance curves demonstrate that MMR variants not only converge faster but also maintain comparable peak performance compared to their baseline counterparts. The consistency of these patterns across different model sizes (1.5B, 7B, 8B), training methods (GRPO, DR-GRPO, DAPO), and benchmarks provides strong evidence for the generality and robustness of the MMR-based reward reweighting approach.

D.2 Exploration Ability: Pass@k Analysis for GRPO and DR-GRPO

Figures 6 and 7 present pass@k curves for GRPO and DR-GRPO methods across all three model scales, complementing the DAPO pass@k analysis shown in the main text (Figure 3).

For GRPO methods (Figure 6), MMR-GRPO and vanilla GRPO exhibit nearly identical pass@k curves across all model sizes and k values. At k=1 (equivalent to majority voting), both methods achieve similar performance. As k increases, the curves remain tightly coupled, with both methods reaching comparable coverage at k=16 (approximately 0.78 for 1.5B, 0.82 for 7B, and 0.82 for 8B). The overlapping curves indicate that MMR-trained GRPO models retain the same diversity of solution strategies as vanilla GRPO models at inference time.

Similarly, for DR-GRPO methods (Figure 7), MMR-DR-GRPO and vanilla DR-GRPO show nearly indistinguishable pass@k performance across all configurations. The curves overlap across the entire range of k values (1-16) for all three model scales, with final coverage at k=16 being virtually identical between MMR and non-MMR variants (approximately 0.80 for 1.5B, 0.84 for 7B, and 0.83 for 8B).

These results, combined with the DAPO pass@k analysis in the main text, provide comprehensive evidence that MMR-based reward reweighting does not compromise the trained model's ability to explore diverse solution strategies during inference. The consistency of this finding across all three training methods (GRPO, DR-GRPO, DAPO) and all three model scales (1.5B, 7B, 8B) demonstrates that diversity-aware training enhances training efficiency without restricting the model's representational flexibility or exploration capabilities.

Configuration	DAPO	GRPO	DR-GRPO
Algorithm-specific Parameters			
Clipping bound (lower) ϵ_{low}	0.2	–	–
Clipping bound (upper) ϵ_{high}	0.28	–	–
KL penalty coefficient β	–	0.04	0.04
Dynamic sampling	Applicable	N/A	N/A
Filter reward index	0 (accuracy)	–	–
Max generation batches	10	–	–
Reward Functions and Weights			
Primary reward	Accuracy (1.0)	Format (1.0)	Format (1.0)
Secondary reward	Cosine (1.0)	Cosine (2.0)	Cosine (2.0)

Table 10: Method-specific configurations for DAPO, GRPO, and DR-GRPO.

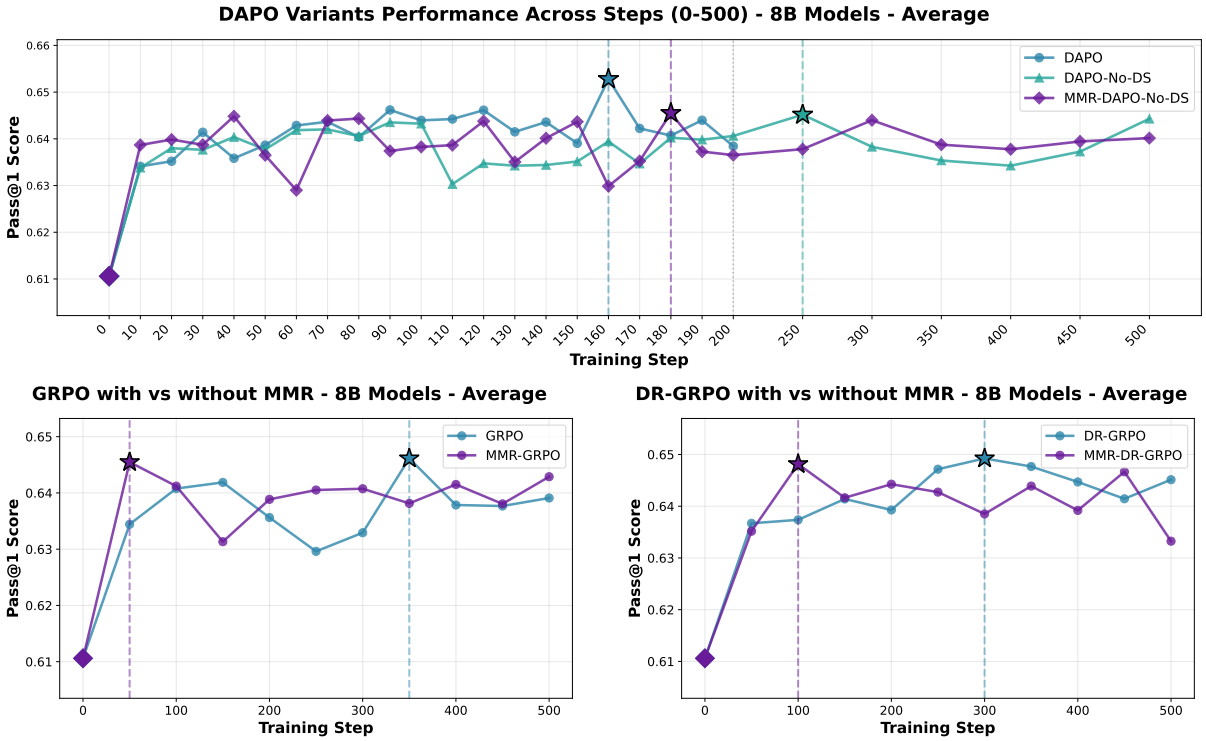

Figure 4: Performance across training steps for 8B models across all three training methods (DAPO, GRPO, DR-GRPO). MMR variants consistently achieve faster convergence and reach peak performance with fewer training steps.

E Performance Across Training Steps on Each Dataset

This section provides a comprehensive analysis of model performance evolution across training steps on all evaluation benchmarks. We present results for three model sizes (1.5B, 7B, 8B) across five mathematical reasoning benchmarks: MATH-500, AIME 2024, Minerva Math, AMC 2023, OlympiadBench, and their average performance.

E.1 GRPO and DR-GRPO Performance Across Training Steps on Each Dataset

Figures 8, 9, and 10 illustrate the pass@1 performance evolution for GRPO method with and without MMR reweighting across 500 training steps. Figures 11, 12, and 13 illustrate the pass@1 performance evolution for DR-GRPO method with and without MMR reweighting across 500 training steps. Each figure contains six subplots corresponding to individual benchmarks and their average.

E.2 DAPO Performance Across Training Steps on Each Dataset

Figures 14, 15, and 16 present DAPO performance across an extended training range (0-500 steps) using a non-linear x-axis scale. The x-axis allocates 60% of space to steps 0-200 (critical training phase) and 40% to steps 200-500 (overfitting phase). A gray dotted vertical line at step 200 marks this

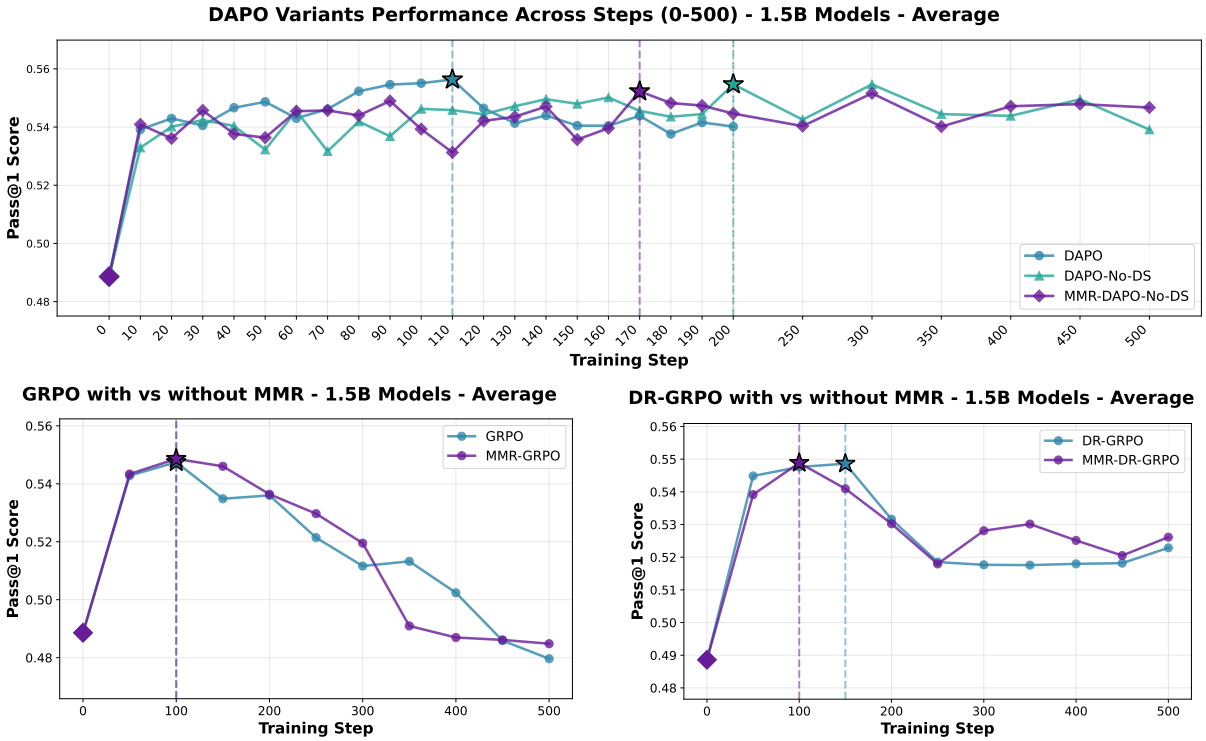

Figure 5: Performance across training steps for 1.5B models across all three training methods (DAPO, GRPO, DR-GRPO). MMR variants consistently achieve faster convergence and reach peak performance with fewer training steps.

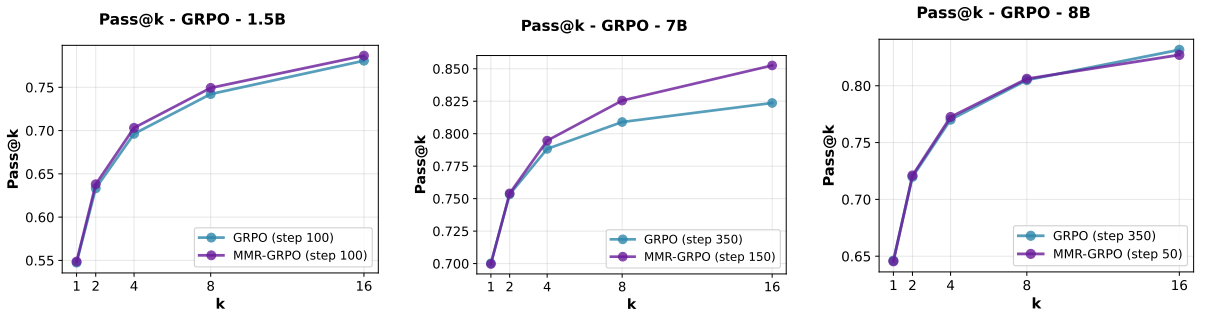

Figure 6: Pass@k curves for GRPO methods across three model scales (1.5B, 7B, 8B). MMR-GRPO maintains nearly identical pass@k performance to vanilla GRPO across all k values.

transition.

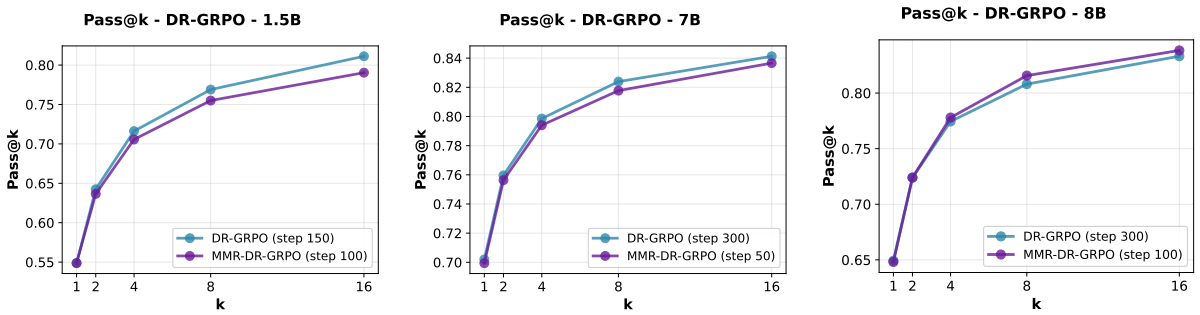

Figure 7: Pass@k curves for DR-GRPO methods across three model scales (1.5B, 7B, 8B). MMR-DR-GRPO shows nearly indistinguishable pass@k performance from vanilla DR-GRPO.

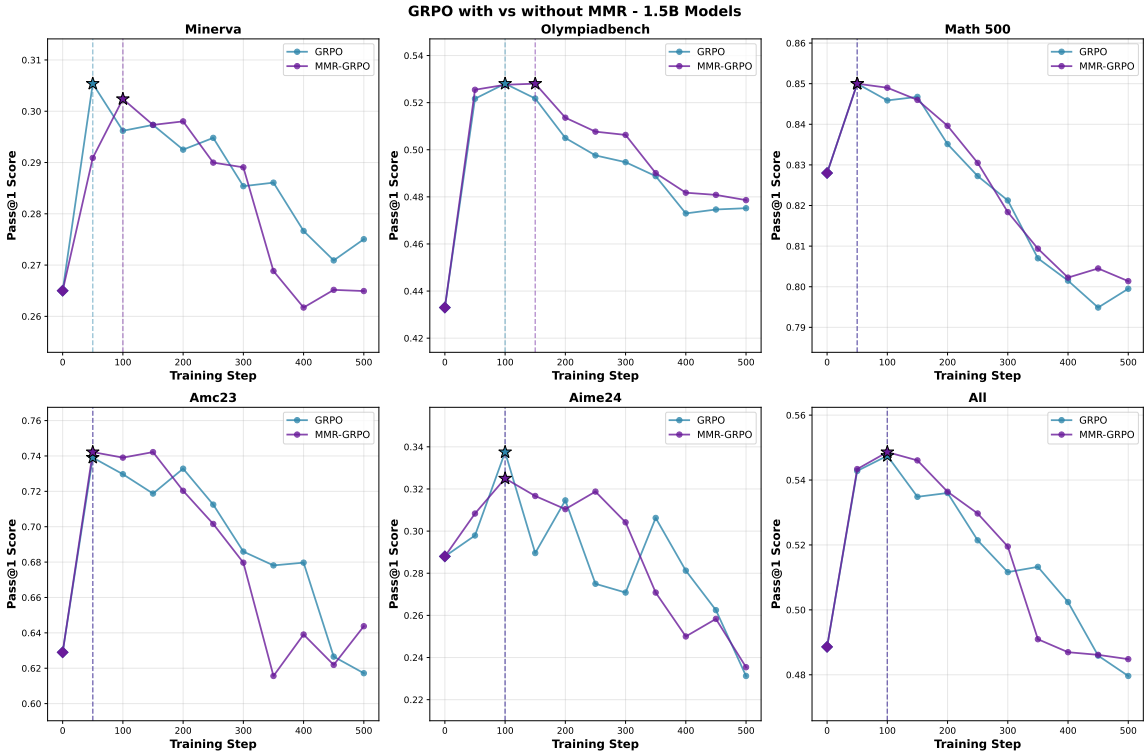

Figure 8: Pass@1 performance across training steps for GRPO with and without MMR reweighting on 1.5B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

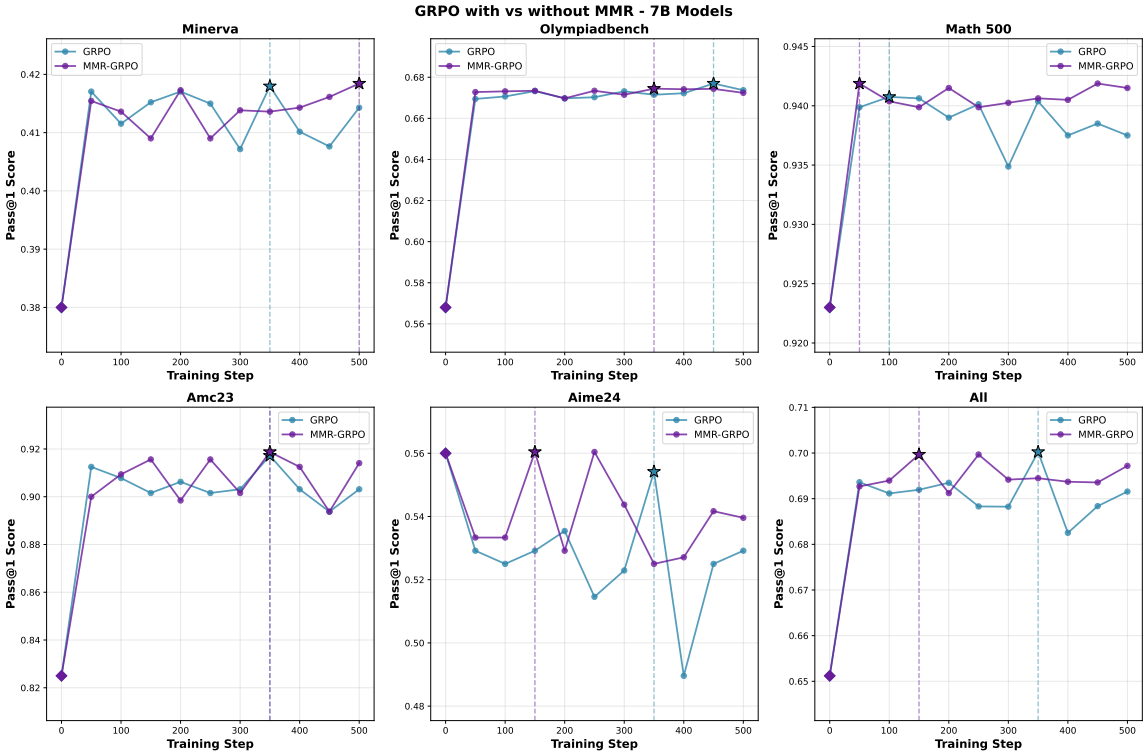

Figure 9: Pass@1 performance across training steps for GRPO with and without MMR reweighting on 7B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

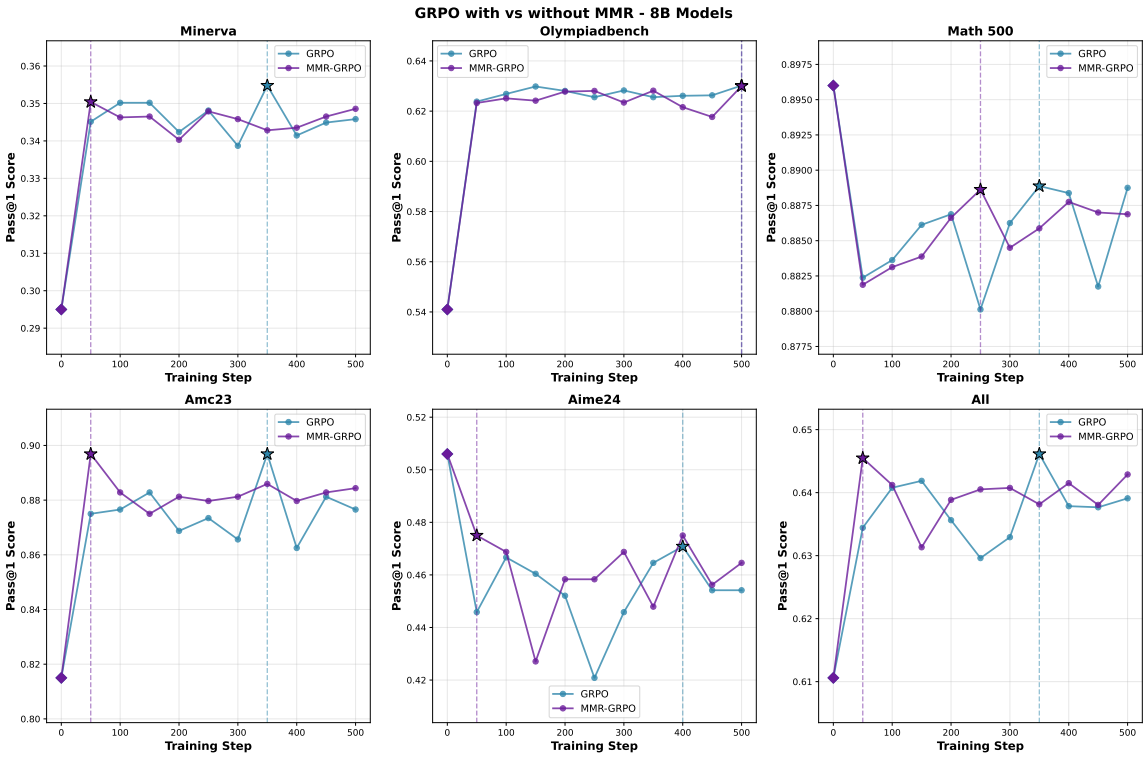

Figure 10: Pass@1 performance across training steps for GRPO with and without MMR reweighting on 8B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

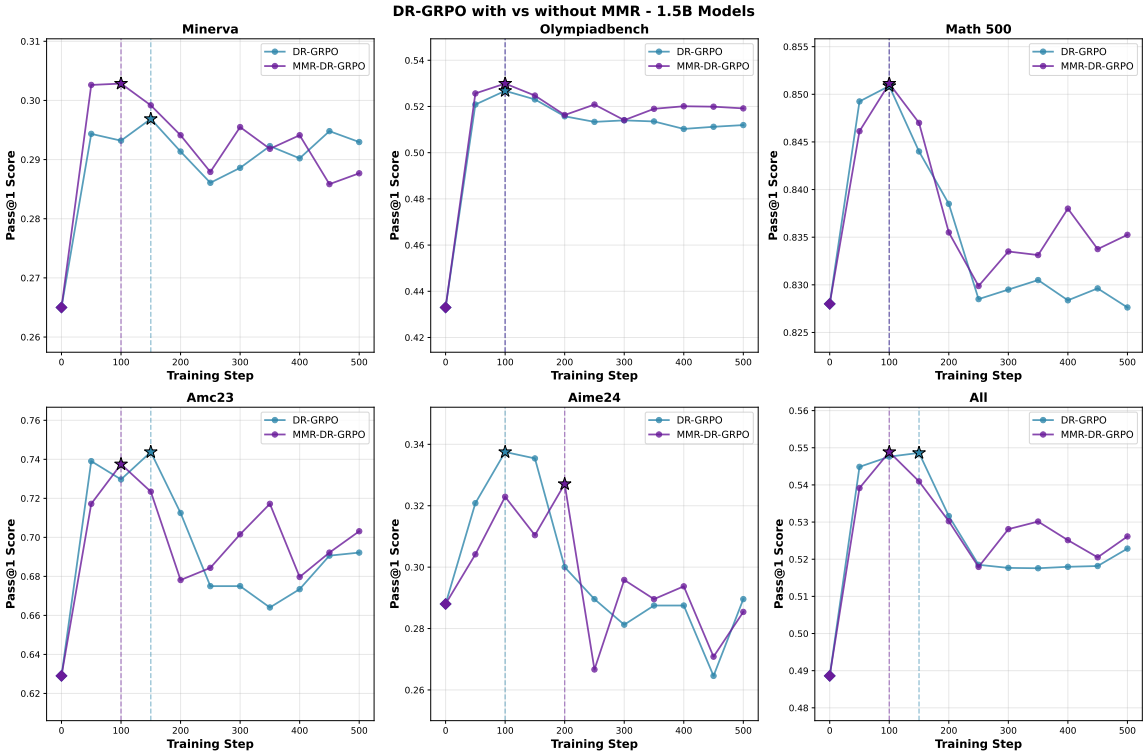

Figure 11: Pass@1 performance across training steps for DR-GRPO with and without MMR reweighting on 1.5B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

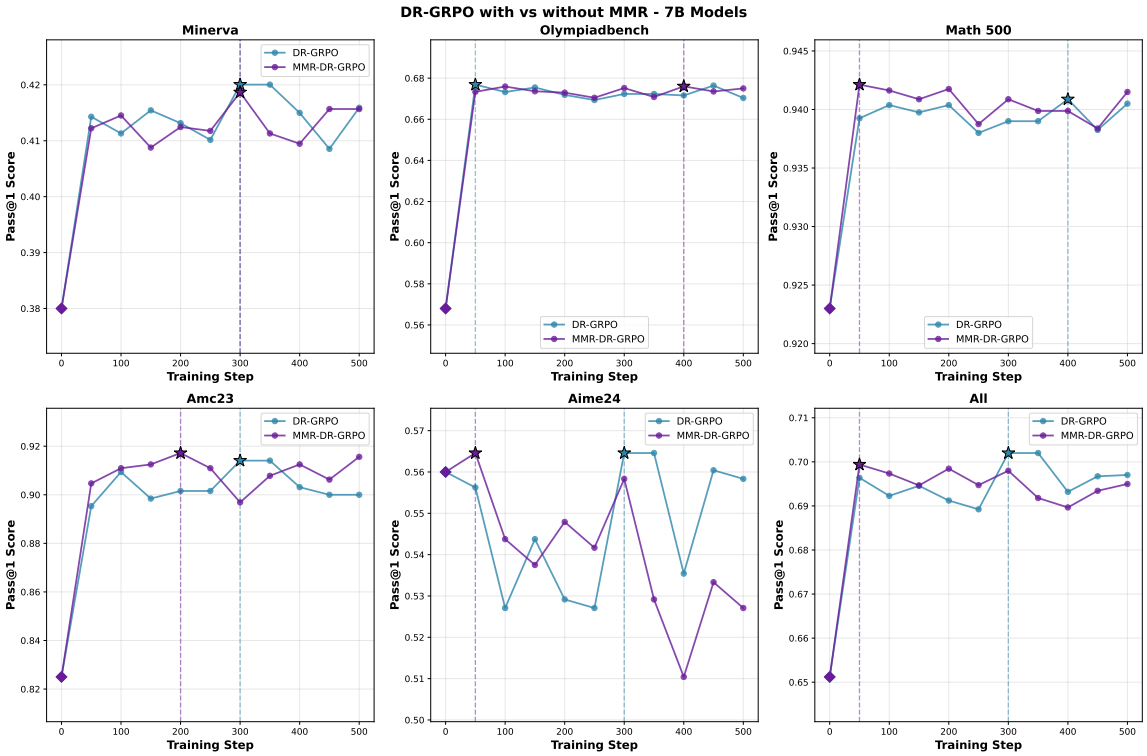

Figure 12: Pass@1 performance across training steps for DR-GRPO with and without MMR reweighting on 7B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

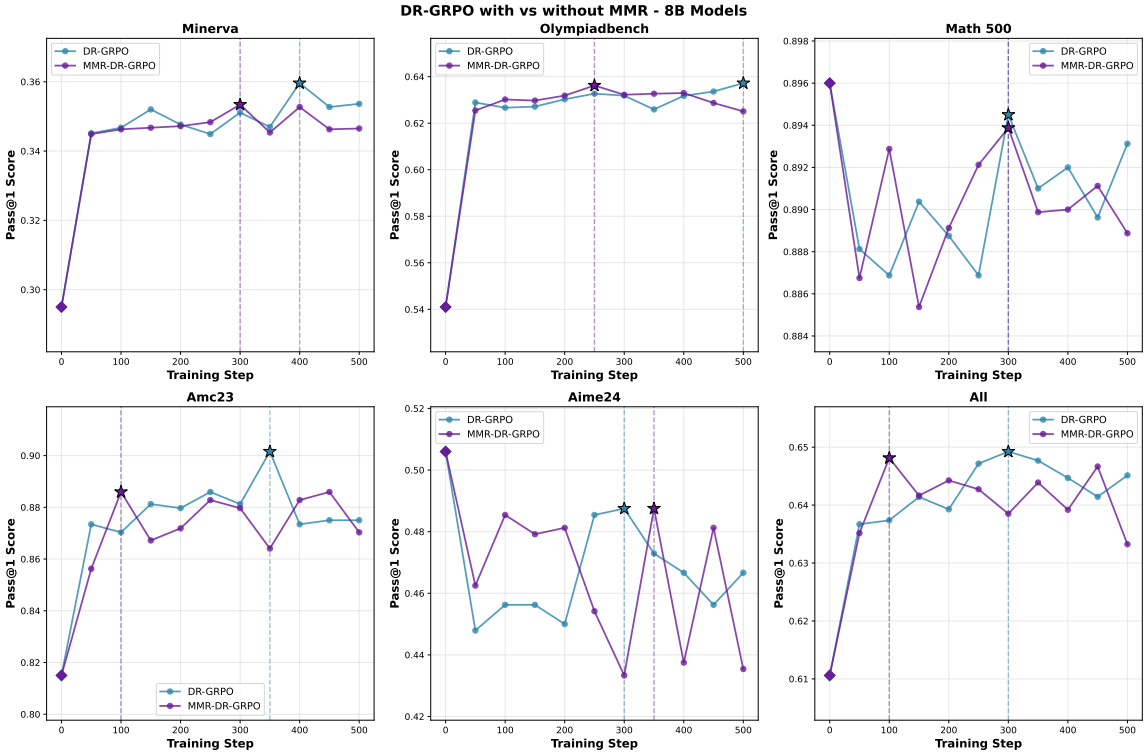

Figure 13: Pass@1 performance across training steps for DR-GRPO with and without MMR reweighting on 8B models. Each subplot shows results for a specific benchmark. Diamond markers indicate baseline performance (step 0), star markers show peak performance, and dashed vertical lines indicate the optimal training step for each configuration.

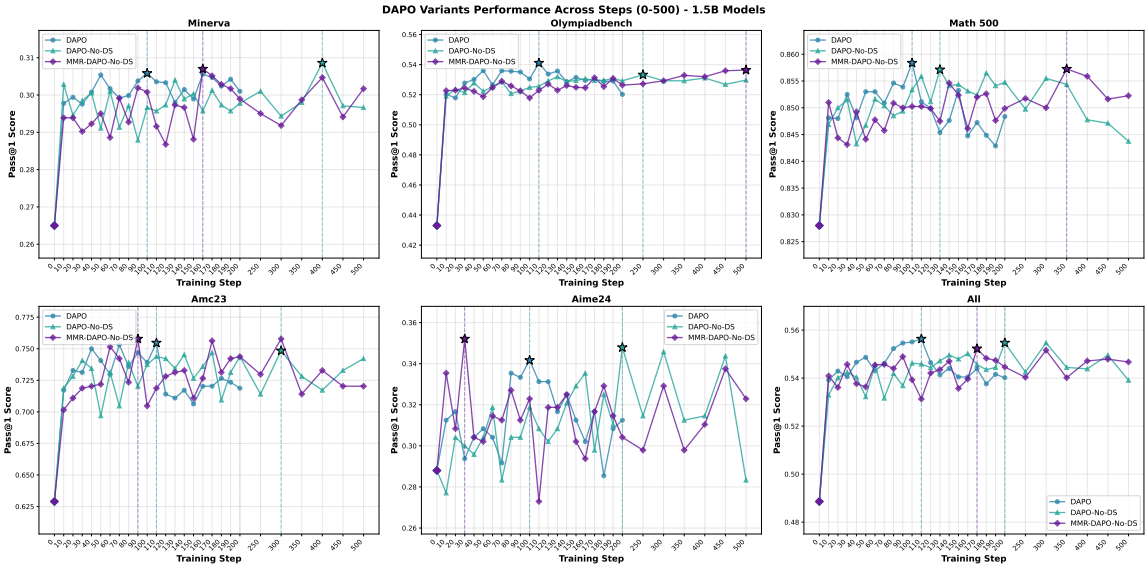

Figure 14: Pass@1 performance across extended training steps (0-500) for DAPO variants on 1.5B models. The x-axis uses non-linear scaling: steps 0-200 occupy 60% of the space, steps 200-500 occupy 40%. The gray dotted line marks step 200. Star markers show peak performance.

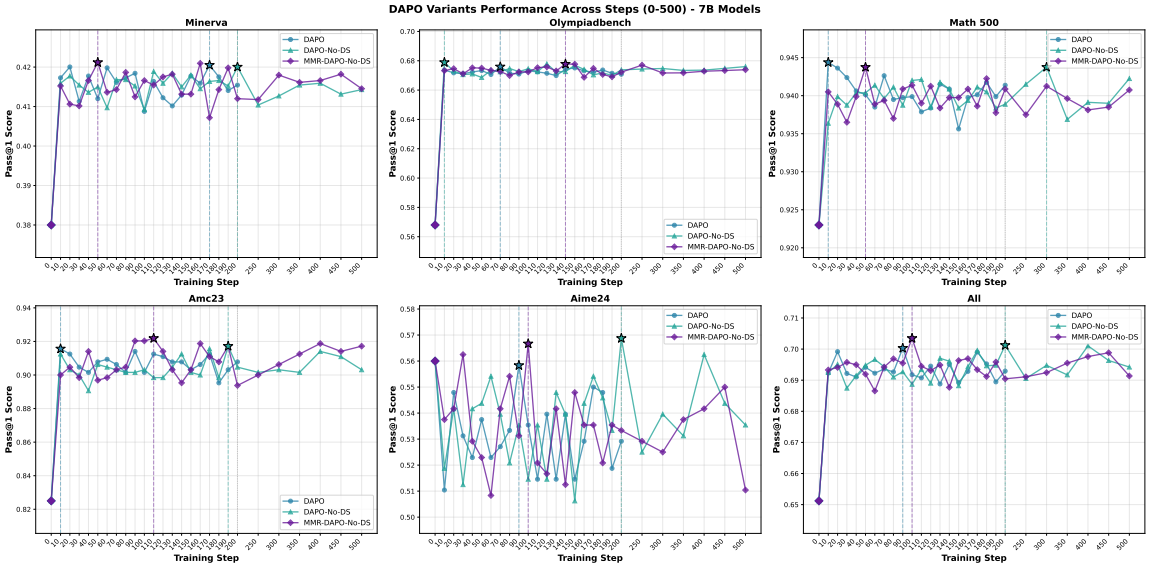

Figure 15: Pass@1 performance across extended training steps (0-500) for DAPO variants on 7B models. The x-axis uses non-linear scaling: steps 0-200 occupy 60% of the space, steps 200-500 occupy 40%. The gray dotted line marks step 200. Star markers show peak performance.

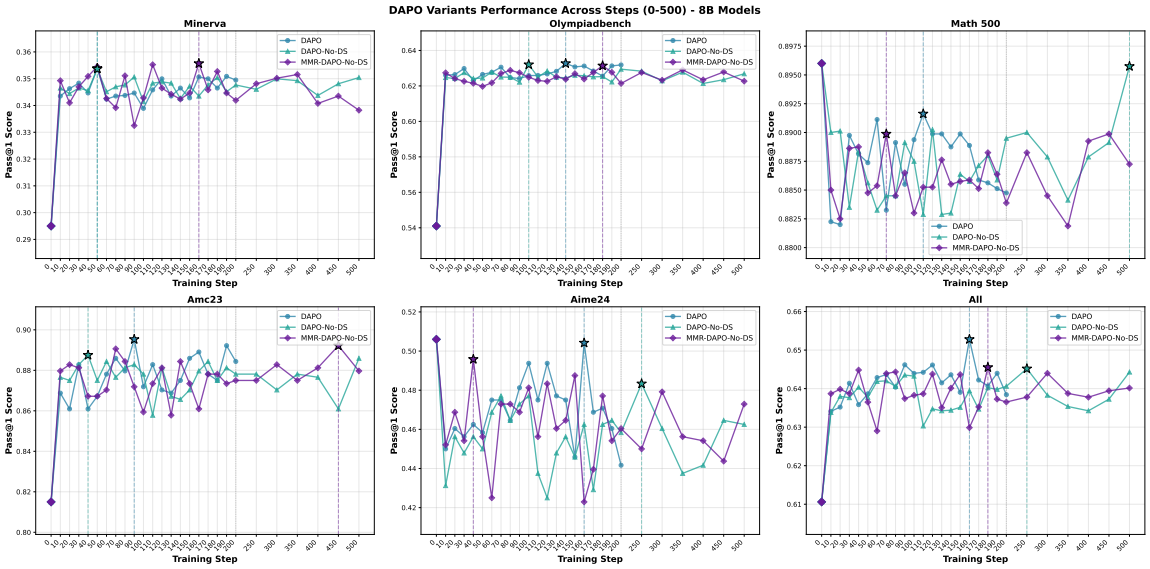

Figure 16: Pass@1 performance across extended training steps (0-500) for DAPO variants on 8B models. The x-axis uses non-linear scaling: steps 0-200 occupy 60% of the space, steps 200-500 occupy 40%. The gray dotted line marks step 200. Star markers show peak performance.