PRAG-PO: PROMPT AUGMENTED POLICY OPTIMIZATION FOR ROBUST AND DIVERSE MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning algorithms such as group-relative policy optimization (GRPO) have shown strong potential for improving the mathematical reasoning capabilities of large language models. While a growing body of work seeks to improve training entropy, rollout diversity, and exploration, most existing methods still train models with a single fixed reasoning prompt or template, which can encourage prompt-specific overfitting and unstable training dynamics. In this work, we introduce Prompt Augmented Policy Optimization (PrAg-PO), a simple policy optimization method that mixes prompt templates with template-specific format rewards during training. By encouraging models to generate reasoning traces under diverse instructions and output formats, PrAg-PO increases rollout diversity and improves robustness. Compared with GRPO and DAPO, PrAg-PO achieves significantly higher reasoning accuracy while mitigating premature training collapse. Empirically, experiments on DeepSeek-R1-Distill-Qwen-1.5B, Qwen2.5-Math-1.5B, Qwen3-1.7B, and Qwen3-8B show that PrAg-PO consistently outperforms strong baselines and achieves competitive performance against recent methods on mathematics benchmarks, using only a fixed MATH Level 3-5 training set of 8.5K problems.

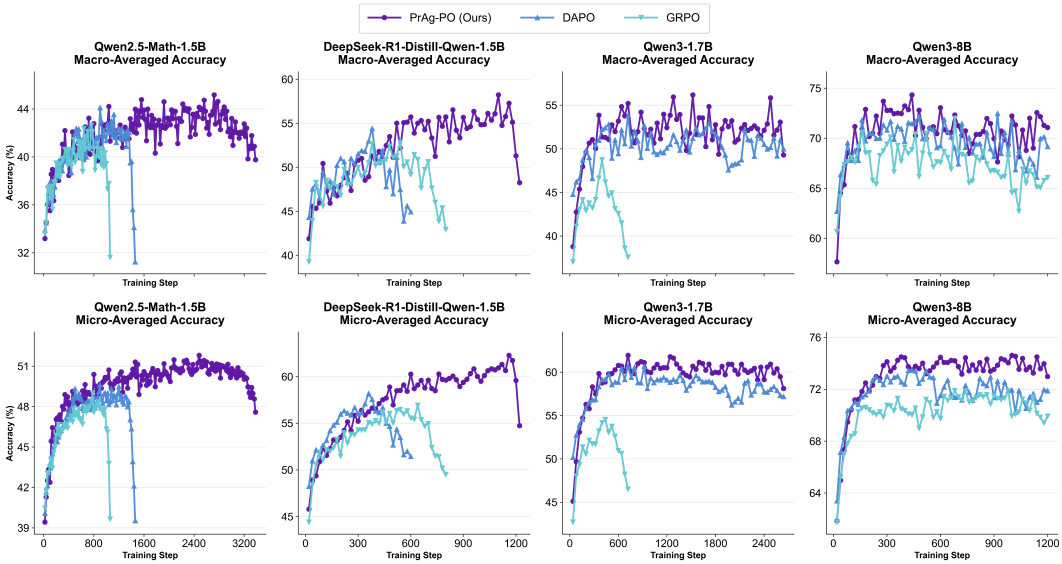

Figure 1: Evaluation accuracy over training steps averaged over mathematics benchmarks for models trained on MATH Level 3-5. PrAg-PO significantly outperforms GRPO and DAPO across Qwen2.5, DeepSeek-R1-Distilled and Qwen3 models, substantially delaying the overfitting and model collapse.

1 INTRODUCTION

Since the introduction of GRPO Shao et al. (2024) and DeepSeek-R1 Guo et al. (2025), which are built upon classical reinforcement learning algorithms such as TRPO Schulman et al. (2015) and PPO Schulman et al. (2017), reinforcement learning (RL) has demonstrated strong capability in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

improving models' reasoning performance on verifiable tasks, including mathematics Yu et al. (2025); Zeng et al. (2025); Zheng et al. (2025); Zhao et al. (2025); Gao et al. (2025), coding Pourreza et al. (2025), medical diagnosis Wang et al. (2025), and tabular reasoning Yang et al. (2025). RL has proven to be an effective post-training paradigm for large language models.

Numerous empirical and theoretical efforts have been made to improve the training objective of the GRPO algorithm. Notably, DAPO Yu et al. (2025), Dr. GRPO Liu et al. (2025d), and GSPO Zheng et al. (2025) unanimously remove the KL loss term from the original GRPO objective, arguing that it may limit reasoning performance by constraining the trained model to remain too close to the reference model. In addition, many works Cui et al. (2025); Cheng et al. (2025); Shen (2025) have proposed a variety of entropy regularization techniques to address entropy collapse phenomenon: a sharp decrease in training entropy as reinforcement learning post-training progresses. Entropy serves as a proxy for policy stochasticity and output diversity. Viewing an LLM as a softmax policy over tokens, entropy collapse corresponds to probability mass concentrating on a narrow subset of tokens, severely limiting the diversity of sampled reasoning trajectories.

Despite the advancements to promote diversity, almost all existing work trains LLMs using a single reasoning format per training run, with a fixed set of format rewards. For example, DeepSeek-R1 Guo et al. (2025) and Open-R1 Hugging Face (2025) adopt a tagged reasoning format (e.g., <think>...</think><answer>...</answer>), while SimpleRL-Zoo Zeng et al. (2025) and OAT-Zero Liu et al. (2025c) rely on Qwen's original chain-of-thought prompting with free-form generation. We hypothesize that such homogeneous reasoning formats encourage overfitting to a single reasoning style, thereby reducing reasoning diversity. Moreover, given the expressive power of chain-of-thought reasoning, there exists a large and largely unexplored design space in which models can reason in fundamentally different ways. Also, given LLM's notorious sensitivity to prompts Yang et al. (2024b); Lu et al. (2025b), optimizing across different prompt formats can improve model's robustness to prompt variations, thereby potentially improve generalization and test-time performance.

In this work, we introduce PrAg-PO, a simple yet effective policy optimization technique for RL post-training of LLMs on mathematical reasoning. Specifically, we mix multiple reasoning templates and formats within a single training run, including tagged reasoning–answer separation, free-form generation, explicit chain-of-thought prompting, and reflection-based formats. These templates are paired with template-specific format rewards to ensure faithful adherence during training. We show that PrAg-PO successfully elicits diverse reasoning behaviors within a single model trained in a single run. Importantly, under a fixed training dataset and the default evaluation prompt template, PrAg-PO achieves competitive performance on mathematical reasoning benchmarks. As shown in Figure 1, our method outperforms both vanilla GRPO and DAPO in terms of per-benchmark mean accuracy and per-question mean accuracy on Qwen2.5-Math-1.5B, DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-1.7B and Qwen3-8B. It is important to note that such improvement is non-trivial, as mixed template training inherently causes distributional difference between train and test time, potentially impairing the performance. We also find that prompt augmentation helps stabilizing RL training, reducing overfitting and allowing continued improvement. In summary, our contributions are three-fold:

1. We propose Prompt Augmented Policy Optimization (PrAg-PO) for RL-based mathematical reasoning, a simple, effective, and computationally inexpensive policy optimization method that employs a diverse set of reasoning templates with associated format rewards to train models to reason. We show that this approach elicits a richer set of reasoning trajectories.

2. Empirically, prompt augmentation stabilizes training dynamics and substantially delays model collapse, enabling longer training horizons. We further interpret prompt augmentation as a gradient regularizer that smooths optimization and reduces overfitting to a single prompt-reward interface.

3. Across different model families, PrAg-PO significantly outperforms GRPO and DAPO, achieving competitive performance that exceeds leading results from prior literature on standard mathematical reasoning benchmarks, as shown in Table 1. **The code is included in the supplementary material. All code and checkpoints will be released.**

2 Related Works

Reinforcement Learning with Verifiable Rewards. DeepSeek-R1 shows that rule-based verifiable rewards using GRPO algorithm can significantly improve the reasoning capability of pre-trained

LLMs. This leads to a surge in a new research paradigm. DAPO Yu et al. (2025) proposes decoupled-clip, dynamic sampling, dropping KL penalty and token-level gradient loss that promotes exploration diversity. Dr. GRPO Liu et al. (2025d) identifies a length bias in original GRPO's objective and proposes removing normalization term. GSPO Zheng et al. (2025) proposed sequence-level importance ratios which notably stabilize RL training for MoE models. GMPO Zhao et al. (2025) optimizes geometric-mean instead of the arithmetic-mean of token-level rewards. MO-GRPO Ichihara et al. (2025) and GDPO Liu et al. (2026) proposes to decouple the normalization of individual rewards in multi-reward settings. Overall, most research centers on enhancing the stability and performance.

Increasing Data and Rollout Diversity for RL. Early seminal works in RL Williams & Peng (1991) have added an entropy regularization term to the policy objective to encourage exploration Mnih et al. (2016); Schulman et al. (2017); Haarnoja et al. (2018). However, it does not result in notable performance gains for LLMs Cui et al. (2025) due to LLM's extremely large response space Shen (2025), which has led to a growing body of work on entropy regularization Cui et al. (2025); Cheng et al. (2025); Shen (2025). Notably, DRA-GRPO Chen et al. (2025b) proposes a diversity-aware reward by leveraging submodular mutual information. Beyond-Pass@1 Liang et al. (2025) shows that problem synthesis and self-play can maintain policy entropy during training. Recent works Li et al. (2025a); Dai et al. (2026); Le et al. (2026) also propose question augmentation, which augments question content via partial answers or reformulations. However, these methods do not focus on augmenting prompt templates. Moreover, question-level augmentation is typically more expensive and requires carefully designed curricula. The concurrent work TemplateRL Wu et al. (2026) proposes problem-adaptive retrieval-based template without relying on format rewards, while we study the effects of diversity-driven template-format augmentation on training dynamics and performance.

LLM's Notorious Sensitivity to Prompts. LLM's performance has been shown to be notoriously sensitive to the prompts. Chain-of-Thought (CoT) prompting has been shown to substantially improve LLM reasoning performance Wei et al. (2022); Lu et al. (2025a). Wei et al. (2022) demonstrate few-shot CoT demonstrations enable complex reasoning, while Kojima et al. (2022) show that LLMs can perform zero-shot reasoning by simply prepending "Let's think step by step" to the prompt. DeepSeek-R1 Guo et al. (2025) adopts zero-shot prompting and explicitly reports that few-shot prompting consistently degrades performance, indicating that RL-trained models are highly sensitive to evaluation prompts. In contrast, Dr. GRPO Liu et al. (2025d) shows that RL training itself is sensitive to the choice of prompt templates. Large language models as optimizer Yang et al. (2024b) show that one can improve mathematical reasoning simply by optimizing test-time prompts.

3 PRAG-PO: PROMPT AUGMENTED POLICY OPTIMIZATION

3.1 PRELIMINARY: GROUP-RELATIVE POLICY OPTIMIZATION

Group Relative Policy Optimization (GRPO) is a variant of proximal policy optimization (PPO) algorithm that replaces value-function baselines with group-relative normalization. For each question q drawn from a training distribution, GRPO samples a group of rollouts $\{o_1, o_2, ..., o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, and maximizes the following surrogate objective:

$$\mathcal{L}_{\text{GRPO}_{\text{token}}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\right. \tag{1}$$

$$\left.\left(\min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_{i,t}\Big) - \beta D_{KL}^{i,t}\right)\right], \tag{2}$$

Where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}$$

Instead of using sample-level loss proposed in the original GRPO formulation, we here express token-level policy-gradient loss, which is widely adopted in most RL libraries (e.g., TRL von Werra et al. (2020), verl Sheng et al. (2024) and slime Zhu et al. (2025)), leading to fairer credit assignment especially in long-CoT scenarios.

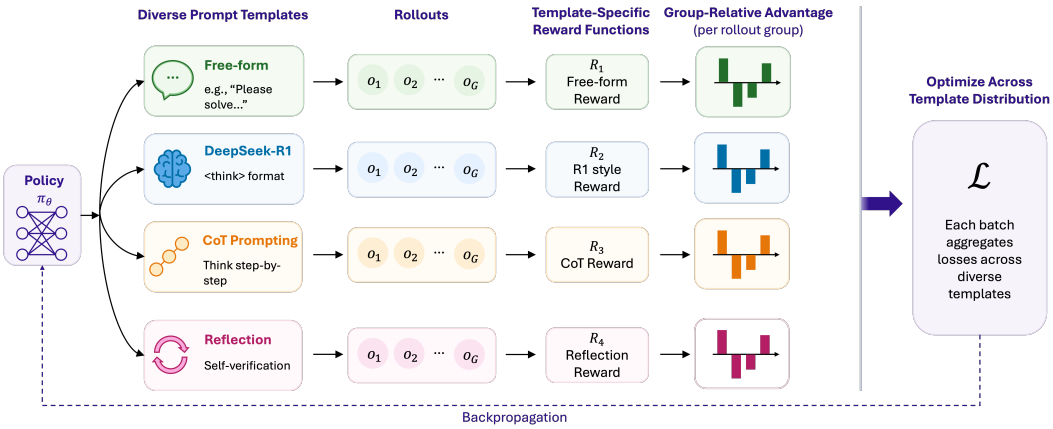

Figure 2: Outline of Prompt Augmented Policy Optimization. The policy is conditioned on diverse prompt templates to generate grouped rollouts. For each group, a template-specific reward function is applied and group-relative advantages are computed. Each batch update therefore aggregates losses across diverse prompt templates.

3.2 Prompt Augmentation

Here we introduce the formulation of prompt augmented policy optimization. Given a set of template functions $\{T_1, T_2, ..., T_K\}$ and corresponding reward functions $\{R_1, R_2, ..., R_K\}$, each template function contains unique prompts and instructions that instruct the model to reason in a certain format. At training time, for each question q drawn from a training distribution, we also uniformly sample a template function and apply it to the question q. We thus maximize the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}},k\sim\text{Uniform}(\{1,...K\})}\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\right. \tag{3}$$

$$\left.\sum_{t=1}^{|o_i|}\min\left(r_{i,t}^{k}(\theta)\hat{A}_{i,t}^{k},\ \text{clip}\left(r_{i,t}^{k}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}}\right)\hat{A}_{i,t}^{k}\right)\right], \tag{4}$$

where

$$r_{i,t}^{k}(\theta) = \frac{\pi_\theta(o_{i,t}|T_k(q),o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|T_k(q),o_{i,<t})}, \qquad \hat{A}_{i,t}^{k} = \frac{R_k(o_i) - \text{mean}(\{R_k(o_i)\}_{i=1}^{G})}{\text{std}(\{R_k(o_i)\}_{i=1}^{G})}$$

Note that we drop the KL penalty as a common practice Yu et al. (2025); Liu et al. (2025d), and use decoupled clipping for entropy regularization. Using the above objective with prompt augmentation, we can elicit the language models to reason in diverse formats in a single training run. Moreover, each group of rollouts has the same template, which allows valid computation of advantage. As each batch contains multiple groups, each update still incorporates gradients from different templates, as shown in Figure 2.

Optimization View: Prompt Augmentation as Gradient Regularizer. From an optimization perspective, prompt augmentation can be viewed as optimizing a prompt-perturbed training objective that regularizes policy optimization under the fixed training question distribution. Let T_k denote a prompt template sampled from $P(T)$, where $P = \text{Uniform}(\{1, \ldots, K\})$. For each template, define the template-conditioned population gradient as

$$G_k(\theta) = \mathbb{E}_{q\sim\mathcal{D}}\left[\nabla_\theta \mathcal{L}_{T_k}(q;\theta)\right].$$

PrAg-PO optimizes the augmented objective whose population gradient is

$$G_{\text{aug}}(\theta) = \mathbb{E}_{k\sim P(T)}[G_k(\theta)].$$

Although evaluation is conducted under a single default prompt template, optimizing the augmented objective can still improve the default-template policy by acting as a regularizer during training. To see this, decompose the template-conditioned population gradient as

$$G_k(\theta) = G_{\text{shared}}(\theta) + \xi_{\text{temp}}(k;\theta),$$

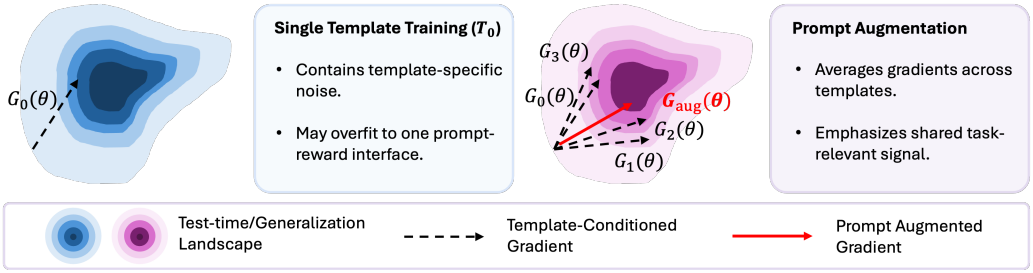

Figure 3: **Prompt augmentation as a gradient regularizer.** Left: Single-template training follows a template-conditioned gradient $G_0(\theta)$, which includes template-specific noise and can lead to overfitting to one prompt–reward interface. Right: Prompt augmentation samples multiple templates and averages their gradients $G_{\text{aug}}(\theta) = \mathbb{E}_{k\sim P(T)}[G_k(\theta)]$, attenuating template-specific components while preserving shared task-relevant signals, resulting in improved generalization.

where $G_{\text{shared}}(\theta)$ denotes gradient components that consistently improve mathematical reasoning across prompt-reward interfaces, while $\xi_{\text{temp}}(k;\theta)$ captures template-specific components induced by prompt wording, output structure, and format-reward design.

As illustrated in Figure 3, a single-template method repeatedly follows $G_0(\theta)$, and may therefore over-optimize template-specific components associated with the default training prompt. Such components can reduce stability and lead to brittle reasoning behavior. In contrast, PrAg-PO averages updates over multiple prompt-reward interfaces. Under the idealized case where template-specific components are zero-mean across the prompt distribution, $\mathbb{E}_{k\sim P(T)}[\xi_{\text{temp}}(k;\theta)] \approx 0$, the augmented gradient reduces to the shared reasoning gradient, $G_{\text{aug}}(\theta) \approx G_{\text{shared}}(\theta)$. In practice, this cancellation need not be exact. As long as template-specific components are less consistently aligned across templates than the shared reasoning components, averaging over prompt-reward interfaces attenuates prompt-idiosyncratic directions relative to a single-template update. Thus, the prompt-augmented objective acts as a training-time regularizer, promoting more robust and generalizable gradients. This is consistent with our empirical results in Figure 1, where PrAg-PO delays training collapse, and with the entropy analysis in Appendix A.3, where it exhibits smoother token-level policy entropy.

3.3 Prompt Template Curation and Selection

We curate a diverse range of reasoning templates and formats to encourage the model to reason differently. Appendix B illustrates examples of templates. Generally, the templates can be divided into four categories. The first group is the DeepSeek Style template that enforces the model to put reasoning processes within special tags like <think> tags and <answer> tags. Such templates are widely used in current literature and open-source projects Hugging Face (2025); Yang et al. (2025). The second group is the free-form reasoning generation, which is Qwen's default style template. The model is instructed to output reasoning processes, and no format constraints are imposed except the final answer format. This style of template is also widely adopted in Qwen native training scenarios Zeng et al. (2025); Liu et al. (2025d). The third group is reflection-based template. The model is instructed to check its work, identify any potential error, and put the verification processes within <check> tags. This form of reasoning is relatively underexplored in current literature, as most work investigated the natural emergence of "aha", "wait" or reflection moments rather than enforce an explicit format for reflection. The fourth group conditions the model on explicit chain-of-thought prompting. The models are conditioned on phrases like "Let's think step by step", and then continue the generation. For some templates with special tags, we additionally include a teacher-forced variant, where the assistant generation is initialized with the corresponding reasoning tags (e.g., <think> or <solution>). This provides stronger structural guidance and improves convergence speed.

Initially, we collect 13 prompt templates from open-source models and evaluation frameworks, including Qwen Yang et al. (2024a), DeepSeek-R1 Guo et al. (2025), Open-R1 Hugging Face (2025), and lm-evaluation-harness Gao et al. (2024). We use these templates to train Qwen2.5-Math-1.5B, Qwen3-1.7B, and Qwen3-8B. In addition, we find that DeepSeek-R1-Distill-Qwen-1.5B benefits substantially from greater template diversity. Therefore, inspired by Toolformer Schick et al. (2023), we leverage the in-context learning capability of a proprietary LLM OpenAI (2026) to expand the template pool to 35 prompt templates for training the distilled model, as detailed in Appendix B.3.

Table 1: Evaluation results on mathematics benchmarks for methods fine-tuned from Qwen2.5-Math-1.5B, DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-1.7B, Qwen3-8B with RL. We report two PrAg-PO checkpoints with the highest average benchmark accuracy and average question accuracy.

Method	**AIME24**	**AMC(23&24)**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
Qwen2.5-Math-1.5B	16.7	43.4	61.8	15.1	28.4	33.1	37.4
GRPO	**23.3**	49.9	75.4	26.3	38.3	42.6	48.4
Dr. GRPO	20.0	53.0	74.2	25.7	37.6	42.1	47.7
SEED GRPO	**23.3**	50.6	75.4	26.8	**41.3**	43.5	49.8
GMPO	20.0	53.0	77.6	30.1	38.7	43.9	50.1
DAPO	**23.3**	**54.9**	76.9	26.0	39.4	44.1	49.6
PrAg-PO (Step 2720)	**23.3**	53.5	78.4	**31.3**	39.4	**45.2**	50.9
PrAg-PO (Step 2480)	20.0	49.4	**80.4**	29.7	41.2	44.1	**51.8**

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
DeepSeek-R1-Distill-Qwen-1.5B	28.8	62.9	82.8	26.5	43.3	48.9	53.5
Still-3-1.5B-Preview	32.5	66.7	84.4	29.0	45.4	51.6	55.6
DeepScaleR-1.5B-Preview	43.1	73.6	**87.8**	30.2	50.0	57.0	59.4
Open-RS1	30.0	70.0	83.8	29.0	52.4	53.0	58.6
Open-RS2	30.0	80.0	85.4	30.5	52.4	55.7	59.6
Open-RS3	**46.7**	72.5	84.4	26.8	51.3	56.3	58.3
Dr. GRPO	33.3	80.0	83.4	30.5	52.1	56.0	58.9
DRA-GRPO	36.7	75.0	86.2	32.4	53.0	56.7	60.5
DRA-Dr.GRPO	36.7	**85.0**	85.2	30.5	53.8	**58.2**	60.5
GRPO	30.0	74.0	85.5	28.2	46.5	52.8	56.5
DAPO	33.3	77.5	84.2	26.9	50.0	54.4	57.6
PrAg-PO (Step 1100)	42.0	78.5	87.5	29.3	53.8	**58.2**	60.9
PrAg-PO (Step 1160)	33.3	77.0	87.2	**32.6**	**56.2**	57.3	**62.3**

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
Qwen3-1.7B	3.3	27.5	57.9	19.1	18.2	25.2	32.9
GAC	20.0	54.1	73.6	25.1	28.1	40.2	43.1
RePO	12.3	47.7	59.6	20.2	**63.1**	40.6	52.8
GRPO	16.7	70.0	78.7	33.2	45.1	48.7	54.1
DAPO	26.7	70.0	84.9	32.8	55.1	53.9	60.7
PrAg-PO (Step 1520)	**33.3**	75.0	85.1	**33.6**	53.9	**56.2**	60.7
PrAg-PO (Step 720)	23.3	**76.5**	**87.0**	33.5	55.7	55.2	**62.0**

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
Qwen3-8B	40.0	67.5	82.0	41.2	44.1	55.0	56.6
GRPO	60.0	**95.0**	93.0	46.1	65.8	72.0	71.9
DAPO	66.7	90.0	92.8	44.8	68.2	72.5	72.6
Critique-GRPO	63.3	92.5	92.0	**52.9**	66.8	73.5	73.2
PrAg-PO (Step 440)	**70.0**	**95.0**	94.3	43.1	69.3	**74.3**	73.5
PrAg-PO (Step 380)	63.3	87.5	**95.8**	45.6	**70.1**	72.5	**74.5**

3.4 Template-Specific Reward Functions

The reward for PrAg-PO is composed of an accuracy reward and a format reward, both ranging from 0 to 1. We use binary rewards for accuracy, and follow TreePO Li et al. (2025c), using Math-Verify Kydlíček (2024) and SymPy Meurer et al. (2017) libraries to verify a response by extracting the final answer enclosed within `\boxed{}`. **Template-Specific reward functions are critical to elicit diverse reasoning formats from the model.** We will show in ablation studies that without these specific reward functions, the model will simply ignore the instructions. Moreover, we found that simple string matching and tag-count-based rewards are sufficient to induce the desired reasoning format, without requiring complex regular expressions. As shown in Appendix B, for DeepSeek-style format, format rewards are computed based on the exact match of special tags. Partial rewards are assigned when the model generates a subset of the required tags; missing or duplicated tags receive no reward. Moreover, the ordering of the tags are not examined, as empirically the model can naturally develop correct ordering. The observation that language models are quick format learners are also observed in other literature Xie et al. (2025), however, we are the first to formally explore it in a hybrid format setting. For templates that do not impose format constraints, we assign the format reward a constant value of 1 to ensure a consistent scale across templates. Since the GRPO objective normalizes rewards within each group, this constant has no effect on the advantage estimates.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets. We use the MATH Level 3-5 dataset Zeng et al. (2025) as the training set, which is a subset of 8,523 questions derived from the MATH dataset Hendrycks et al. (2021). This filtered subset contains problems of medium to high difficulty (i.e., levels 3 to 5 on a five-level scale). The MATH dataset consists of problems from mathematics competitions, including AMC 10, AMC 12, and AIME. Our test set comprises a standard suite of benchmarks, including AIME24 (30 questions) Zhang & Math-AI (2024), AMC (83 questions) AI-MO (2024), AMC23 (40 questions), MATH500 (500 questions), Minerva Math (272 questions) Lewkowycz et al. (2022), and OlympiadBench (675 questions) He et al. (2024), covering a broad range of graduate-level and Olympiad-style problems.

Models and Competitors. We use Qwen2.5-Math-1.5B Yang et al. (2024a), DeepSeek-R1-Distill-Qwen-1.5B Guo et al. (2025), Qwen3-1.7B, and Qwen3-8B Team (2025a) as base models, covering different model families and training stages. Qwen2.5-Math-1.5B is continually pretrained from Qwen2.5-1.5B on mathematical corpora. DeepSeek-R1-Distill-Qwen-1.5B is distilled from the large DeepSeek-R1 model. Qwen3-1.7B and Qwen3-8B are latest-generation instruction-tuned models. We reproduce GRPO Shao et al. (2024) and DAPO Yu et al. (2025) baselines using our code base. For GRPO, we adopt token-level loss. KL coefficient β is 0.001. For DAPO, we implement decoupled clip and KL penalty removal, following the core training objective. In addition to GRPO and DAPO, we compare our method against a comprehensive set of state-of-the-art baselines. For Qwen2.5-Math-1.5B, we include Dr. GRPO Liu et al. (2025d), SEED-GRPO Chen et al. (2025a), and GMPO Zhao et al. (2025). For DeepSeek-R1-Distill-Qwen-1.5B, we include Still-3-1.5B-Preview Team (2025b), DeepScaleR-1.5B-Preview Luo et al. (2025), Open-RS Dang & Ngo (2026), Dr. GRPO Liu et al. (2025d), and DRA-GRPO Chen et al. (2025b). For Qwen3-1.7B, we include GAC Xu et al. (2026) and RePO Li et al. (2025b). For Qwen3-8B, we include Critique-GRPO Zhang et al. (2026).

Training and Evaluation Settings. We train our models using the verl framework Sheng et al. (2024). We use a prompt batch size of 128 with a mini-batch size of 32, resulting in four weight updates per batch. The group size is set to 8, so each mini-batch contains 256 rollouts. Following DAPO, we set the clipping parameters to $\epsilon_{\text{high}} = 0.28$ and $\epsilon_{\text{low}} = 0.20$. Detailed hyperparameters and hardware settings are provided in Section A.2. To ensure fair comparison and avoid introducing test-time scaling effects, **we use model tokenizer's single default template during evaluation,** adopting exactly the same evaluation framework and code base as Dr. GRPO, SEED GRPO and GMPO. We evaluate models' performance under greedy decoding (i.e., temperature = 0), and report pass@1 metric. Furthermore, due to inherent stochasticity of vLLM inference engine Kwon et al. (2023), we let all models perform five inference rounds under the same seed, and report the average accuracy. Importantly, due to the extreme imbalanced nature of the test datasets, we report both per-benchmark and per-question average accuracy. Following standard practice in Dr. GRPO and DAPO, the model is evaluated at periodic checkpoints, and the best-performing results are reported.

4.2 PRAG-PO IMPROVES REASONING ACCURACY AND STABILIZES GROUP-RELATIVE POLICY TRAINING

PrAg-PO demonstrates strong performance in both internal comparisons against our GRPO and DAPO baselines and external comparisons against previously reported methods. As shown in Table 1, PrAg-PO consistently outperforms the GRPO and DAPO baselines in both average benchmark accuracy and average question accuracy, indicating that its performance gains are robust across evaluation criteria. Compared with other competitive and state-of-the-art methods, PrAg-PO also achieves leading performance across model families. For Qwen2.5-Math-1.5B, PrAg-PO achieves the best performance under both evaluation metrics. Notably at step 2480, to the best of our knowledge, this is the first instance where a Qwen2.5-Math-1.5B RL-finetuned model surpasses **80%** accuracy on the MATH500 benchmark. For DeepSeek-R1-Distill-Qwen-1.5B, PrAg-PO again achieves the highest average accuracy in both metrics, having a 2-point lead in per-question accuracy, while also obtaining the best score on OlympiadBench and Minerva. For Qwen3-1.7B, PrAg-PO substantially outperforms all competing methods, exceeding the strongest previously reported result (i.e., RePO) by more than 10 points. For Qwen3-8B, PrAg-PO also leads on the majority of benchmarks. These results demonstrate that prompt augmentation is an effective and broadly applicable method for improving the reasoning accuracy of language models under reinforcement learning fine-tuning.

Let $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, where $i = \sqrt{-1}$. What is $f(2022)$?

Qwen–Math Template Output

Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we need to determine the value of $f(2022)$.

First, let's denote the complex numbers $\omega_1 = \frac{-1+i\sqrt{3}}{2}$ and $\omega_2 = \frac{-1-i\sqrt{3}}{2}$. These are the cube roots of unity, specifically $\omega_1 = \omega$ and $\omega_2 = \omega^2$, where $\omega = e^{2\pi i/3}$ and $\omega^2 = e^{-2\pi i/3}$. This means $\omega^3 = 1$ and $(\omega^2)^3 = 1$.

Thus, we have $\omega_1^3 = 1$ and $\omega_2^3 = 1$. Therefore, $\omega_1^{2022} = (\omega_1^3)^{674} = 1^{674} = 1$ and $\omega_2^{2022} = (\omega_2^3)^{674} = 1^{674} = 1$.

Hence, $f(2022) = \omega_1^{2022} + \omega_2^{2022} = 1 + 1 = 2$.

So, the value of $f(2022)$ is 2.

The final answer is $\boxed{2}$.

DeepSeek-Style Template Output

<think>

Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we can denote the complex numbers $\omega = \frac{-1+i\sqrt{3}}{2}$ and $\omega^2 = \frac{-1-i\sqrt{3}}{2}$. These are the non-real cube roots of unity, satisfying $\omega^3 = 1$ and $(\omega^2)^3 = 1$, and also $\omega^2 + \omega + 1 = 0$.

Thus, we have $f(n) = \omega^n + (\omega^2)^n$.

We need to find $f(2022)$. Since $\omega^3 = 1$, we can reduce the exponent modulo 3:

$$2022 \mod 3 = 0.$$

Therefore, $\omega^{2022} = (\omega^3)^{674} = 1^{674} = 1$ and $(\omega^2)^{2022} = (\omega^6)^{337} = (1)^{337} = 1$.

Hence, $f(2022) = \omega^{2022} + (\omega^2)^{2022} = 1 + 1 = 2$.

The value of $f(2022)$ is 2.

</think>

<answer>

The value of $f(2022)$ is $\boxed{2}$.

</answer>

Reflection-Based Template Output

<solution>

Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we can denote the complex numbers $\omega = \frac{-1+i\sqrt{3}}{2}$ and $\bar{\omega} = \frac{-1-i\sqrt{3}}{2}$. These numbers are the non-real cube roots of unity, and they satisfy the property $\omega^3 = 1$ and $\bar{\omega}^3 = 1$.

We need to find $f(2022)$. First, we observe the periodicity of ω and $\bar{\omega}$:

$$\omega^3 = 1 \implies \omega^{2022} = (\omega^3)^{674} = 1^{674} = 1$$
$$\bar{\omega}^3 = 1 \implies \bar{\omega}^{2022} = (\bar{\omega}^3)^{674} = 1^{674} = 1$$

Thus, we have:

$$f(2022) = \omega^{2022} + \bar{\omega}^{2022} = 1 + 1 = 2$$

Therefore, the value of $f(2022)$ is 2.

The final answer is $\boxed{2}$.

</solution>

<check>

Let's verify step by step:

1. We calculated that $\omega^{2022} = 1$ and $\bar{\omega}^{2022} = 1$.
2. Therefore, $f(2022) = \omega^{2022} + \bar{\omega}^{2022} = 1 + 1 = 2$.

The calculation is correct. The value of $f(2022)$ is indeed 2.

</check>

Figure 4: Reasoning trajectories produced by the model trained with PrAg-PO when answering the same question under different prompt templates. The model generates diverse reasoning formats.

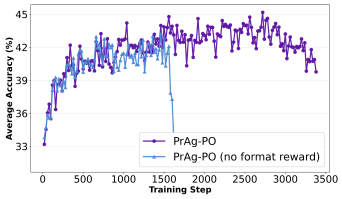

Figure 5: Accuracy of PrAg-PO with and without format rewards across training steps on Qwen2.5-Math-1.5B. Without format rewards, the model collapses earlier.

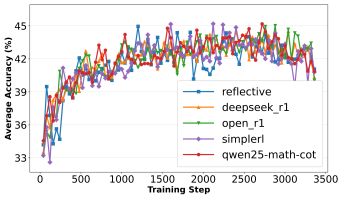

Figure 6: Accuracy of PrAg-PO evaluated with different prompt templates on Qwen2.5-Math-1.5B. Performance is consistent and robust across prompt templates, with no clear outlier.

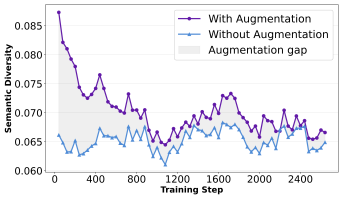

Figure 7: Semantic diversity comparison between Qwen3-1.7B rollouts with augmentation and without augmentation. Augmented rollouts consistently achieve higher diversity.

Training collapse during prolonged GRPO training has been observed in numerous prior studies Dang & Ngo (2026); Qi et al. (2025); Liu et al. (2025a;b). Although its exact causes remain an active area of research, recent work has attributed such instability to factors such as entropy collapse and training–inference precision mismatch. Empirically, we find that PrAg-PO is effective in stabilizing training dynamics and delaying collapse. As shown by 1.5B models in Figure 1, GRPO and DAPO collapse after roughly 400–1500 steps. In contrast, PrAg-PO continues to improve over a substantially longer training horizon, reaching more than 2000 steps. This effect is also reflected in token-level policy entropy, where PrAg-PO exhibits smoother and more stable entropy trajectories compared to DAPO (See Appendix A.3). This improved stability may be driven by the regularizing effect of prompt augmentation. By exposing the policy to diverse prompt and reward interfaces, PrAg-PO reduces overfitting to any single reasoning template. Aggregating losses across different templates smooths the gradient signal, attenuating template-specific noise and reducing abrupt spikes.

4.3 PrAg-PO Elicits Diverse Reasoning Trajectories

In this section, we demonstrate that PrAg-PO enables the model to develop diverse reasoning formats and trajectories. As shown in Figure 4, we evaluate a single trained checkpoint on the same mathematical question, while varying only the prompt template. Despite identical inputs and model parameters, the model adapts its output to follow the instructions imposed by each template, producing distinct reasoning structures. In the left column, the Qwen-Math free-form generation template is applied, resulting in a generic, unconstrained reasoning trace. In the middle column, a DeepSeek-style template instructs the model to place its reasoning within <think>tags and its final answer within <answer>tags. The model successfully adheres to these structural constraints, with the required tags highlighted in blue. In the right column, a reflection-based template is used, prompting the model to include an additional verification step enclosed in <check>tags, which it

also follows correctly. Although the three solutions are mathematically similar, they exhibit subtle yet meaningful differences in reasoning emphasis and presentation. For instance, the left-column output explicitly converts the complex numbers into their exponential form to invoke the cube roots of unity, whereas the middle-column output emphasizes reducing the exponent $2022 \bmod 3$ to simplify the expression. These observations illustrate that prompt augmentation elicits multiple valid reasoning trajectories from a single model, diversifying its outputs to support GRPO training.

In Figure 7, we further quantify the diversity introduced by the augmentation based on test-set rollout semantic diversity of the PrAg-PO Qwen3-1.7B checkpoints under two settings: one using a fixed prompt template, and one using augmented templates. We compute the pairwise cosine distances between rollout embeddings within each setting and compare the mean distances. In detail, we use a temperature of 1.0, and a top_p of 1.0 and generate 8 rollouts per question on the evaluation set, matching the exact hyperparameter settings used during training. The semantic diversity is computed by passing rollouts through the BGE-M3 embedding model Chen et al. (2024), and using (1 - pairwise cosine similarity) as the distance metric:

$$\text{Semantic Diversity} = 1 - \frac{\sum_{0 \le i < j < 8} \cos(E(r_i), E(r_j))}{8C_2} \tag{5}$$

where r_i is i-th rollout and E is the embedding model. Figure 7 shows that the semantic diversity of rollouts with augmentation is consistently higher than that without augmentation over almost all checkpoints. The gap is particularly large in the initial-to-mid stage of training, demonstrating that PrAg-PO promotes substantially more diverse exploration during early stages of policy optimization.

4.4 Ablation Studies

Template-Specific Format Rewards Are Critical for Stable Training. In this section, we examine whether template-specific format rewards are necessary for improving reasoning performance and eliciting diverse reasoning trajectories. In principle, these benefits could be attributed solely to prompt-level data augmentation, without modifying the training objective. To isolate the effect of format rewards, we remove them entirely and train the model using only a binary accuracy reward, while still applying the augmented prompt templates. The resulting accuracy curves are shown as the light blue line in Figure 5. Without format rewards, training collapses at approximately 1,500 steps, and the model consistently underperforms the counterpart trained with format rewards. Through manual inspection of the model's rollouts, we observe that under this setting the model largely ignores the prompt instructions and defaults to free-form reasoning outputs. This behavior is expected, as in the absence of explicit reward signals, the model has no incentive to follow diverse reasoning formats.

There is No "Secret" Gold Prompt. A natural concern is that the effectiveness of PrAg-PO may be driven by the accidental inclusion of a "gold" prompt that substantially boosts training performance. In this section, we show that this is not the case: after prompt-augmented policy optimization, the model performs comparably across different prompt templates, indicating that no single template provides a disproportionate advantage. We evaluate Qwen2.5-Math-1.5B at periodic checkpoints under different prompt templates. As shown in Figure 6, accuracies are largely consistent across templates, fluctuating around a common mean without clear outliers. This suggests that the model becomes robust to prompt variation, and that the performance gains arise from the collective effect of prompt augmentation rather than from a single hidden "gold" prompt. In Section A.5, we further evaluate models on **out-of-distribution prompts**, and shows PrAg-PO has the smallest performance degradation comparing to DAPO and GRPO, which reinforces its robustness and generalization.

5 Conclusion, Limitations and Future Works

We introduce Prompt Augmented Policy Optimization (PrAg-PO) for training LLMs on mathematical reasoning tasks. We show that prompt augmentation, when combined with template-specific format rewards, elicits diverse reasoning trajectories, stabilizes training, and improves reasoning accuracy. One limitation is that PrAg-PO does not fully eliminate long-horizon optimization instability, as the model may still collapse after sufficiently prolonged training. Nevertheless, under a fixed dataset budget, PrAg-PO enables more stable training dynamics and achieves stronger performance. Another limitation is that the template pool still involves human design choices. More systematic template selection could improve scalability, although our analysis suggests that PrAg-PO does not rely on a single "gold" prompt. As a promising direction for future work, the availability of multiple reasoning formats opens the door to inference-time scaling strategies that may further enhance performance.

AI USE STATEMENT

In this work, we used generative AI tools for generate synthetic data sets and support qualitative and thematic data analysis. We have not used generative AI tools for help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, interpret results, and formulate mathematical claims, provide critical ingredients for proving mathematical claims, and assist in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for edit a research paper to improve readability. We have reviewed all AI-assisted work. For example, all LLM edits on writing are verified by authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

AI-MO. AIMO Validation AMC Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc, 2024. Accessed: 2026-05-06.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

Minghan Chen, Guikun Chen, Wenguan Wang, and Yi Yang. Seed-grpo: Semantic entropy enhanced grpo for uncertainty-aware policy optimization. *arXiv preprint arXiv:2505.12346*, 2025a.

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

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. Harder is better: Boosting mathematical reasoning via difficulty-aware grpo and multi-aspect question reformulation. *arXiv preprint arXiv:2601.20614*, 2026.

Quy-Anh Dang and Chris Ngo. Reinforcement learning for reasoning in small LLMs: What works and what doesn't. In *Logical and Symbolic Reasoning in Language Models @ AAAI 2026*, 2026. URL https://openreview.net/forum?id=3pWL6Zxc4A.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 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)*, pp. 3828–3850, 2024.

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

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. Mo-grpo: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *arXiv preprint arXiv:2509.22047*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Hynek Kydlíček. Math-Verify: Math verification library, 2024. URL https://github.com/huggingface/math-verify.

Khiem Le, Youssef Mroueh, Phuc Nguyen, Chi-Heng Lin, Shangqian Gao, Ting Hua, and Nitesh V Chawla. Transform-augmented grpo improves pass@ k. *arXiv preprint arXiv:2601.22478*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. *arXiv preprint arXiv:2507.13266*, 2025a.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025b.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, et al. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*, 2025c.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*, 2025.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Zhuo Jiang. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025a. URL https://richardli.xyz/rl-collapse.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=YPsJha5HXQ.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. There may not be aha moment in r1-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero, 2025c. Notion Blog.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025d. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025a.

Wenquan Lu, Jiaqi Zhang, Hugues Van Assel, and Randall Balestriero. Ditch the denoiser: Emergence of noise robustness in self-supervised learning from data curriculum. In *Advances in Neural Information Processing Systems*, volume 38, pp. 136057–136089, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c6a4fc50b10ec1add5086e2f527ff0d3-Paper-Conference.pdf.

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

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

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

OpenAI. Chatgpt. https://chat.openai.com/, 2026. Accessed: 2026-03-20.

Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan O Arik. Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=HbwkIDWQgN.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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

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

Han Shen. On entropy control in llm-rl algorithms. *arXiv preprint arXiv:2509.03493*, 2025.

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

Qwen Team. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

RUCAIBox STILL Team. Still-3-1.5b-preview: Enhancing slow thinking abilities of small models through reinforcement learning. 2025b. URL https://github.com/RUCAIBox/Slow_Thinking_with_LLMs.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Hanyin Wang, Zhenbang Wu, Gururaj J. Kolar, Hariprasad Reddy Korsapati, Brian Bartlett, Bryan Hull, and Jimeng Sun. Reinforcement learning for out-of-distribution reasoning in LLMs: An empirical study on diagnosis-related group coding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0jvnfH0WYV.

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

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Jinyang Wu, Chonghua Liao, Mingkuan Feng, Shuai Zhang, Zhengqi Wen, Haoran Luo, Ling Yang, Huazhe Xu, and Jianhua Tao. TemplateRL: Structured template-guided reinforcement learning for LLM reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4050–4080, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.198. URL https://aclanthology.org/2026.findings-acl.198/.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025.

Haofeng Xu, Junwei Su, Yukun Tian, Lansong Diao, Zhengping Qian, and Chuan Wu. Gac: Stabilizing asynchronous rl training for llms via gradient alignment control. *arXiv preprint arXiv:2603.01501*, 2026.

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

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=Bb4VGOWELI.

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. Table-r1: Inference-time scaling for table reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20616–20635, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun MA, and Junxian He. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=vSMCBUgrQj.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen M. Meng. Advancing LLM reasoning with natural language and numerical feedback. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gz7hVnrRWq.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, et al. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

Outline of the Appendix:

- Section A: Further Experiment Results
  - Section A.1: Evaluation Results with Standard Deviation
  - Section A.2: Hyperparameters and Hardware Settings
  - Section A.3: Does Prompt Augmentation Increase Token-level Policy Entropy?
  - Section A.4: Increasing KL Regularization Does Not Match PrAg-PO
  - Section A.5: Out-Of-Distribution Template Evaluation
- Section B: Prompts and Format Reward Details
  - Section B.1: Template Statistics Visualization
  - Section B.2: 13 Templates for Qwen2.5 and Qwen3
  - Section B.3: 35 Templates for DeepSeek-R1
  - Section B.4: Format Reward Best Practices

A Further Experiment Results

A.1 Evaluation Results with Standard Deviation

Table 2: Evaluation statistics for Table 1 Qwen2.5-Math-1.5B with standard deviation over the 5 inference rounds.

Method	**AIME24**	**AMC**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	23.33 ± 0.00	49.88 ± 1.37	75.36 ± 0.79	26.32 ± 0.49	38.28 ± 0.31	42.64 ± 0.40	48.41 ± 0.42
DAPO	23.33 ± 0.00	54.94 ± 1.37	76.92 ± 0.54	25.96 ± 0.56	39.44 ± 0.32	44.12 ± 0.37	49.62 ± 0.33
PrAg-PO (Step 2720)	23.33 ± 0.00	53.49 ± 1.08	78.40 ± 0.45	31.25 ± 0.37	39.41 ± 0.66	45.18 ± 0.19	50.92 ± 0.30
PrAg-PO (Step 2480)	20.00 ± 0.00	49.40 ± 1.20	80.40 ± 0.32	29.71 ± 0.48	41.24 ± 0.59	44.15 ± 0.32	51.81 ± 0.28

Table 3: Evaluation statistics for Table 1 DeepSeek-R1-Distill-Qwen-1.5B with standard deviation over the 5 inference rounds..

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	30.00 ± 0.00	74.00 ± 1.37	85.48 ± 0.18	28.24 ± 0.71	46.49 ± 0.12	52.84 ± 0.28	56.47 ± 0.14
DAPO	33.33 ± 0.00	77.50 ± 1.77	84.24 ± 0.97	26.91 ± 1.57	50.04 ± 0.91	54.41 ± 0.34	57.56 ± 0.46
PrAg-PO (Step 1100)	42.00 ± 2.98	78.50 ± 3.35	87.48 ± 0.73	29.34 ± 0.92	53.81 ± 0.64	58.23 ± 1.10	60.94 ± 0.18
PrAg-PO (Step 1160)	33.33 ± 0.00	77.00 ± 2.74	87.24 ± 1.05	32.65 ± 0.16	56.18 ± 0.83	57.28 ± 0.84	62.29 ± 0.63

Table 4: Evaluation statistics for Table 1 Qwen3-1.7B with standard deviation over the 5 inference rounds..

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	16.67 ± 0.00	70.00 ± 0.00	78.72 ± 0.87	33.16 ± 0.55	45.13 ± 0.37	48.73 ± 0.13	54.15 ± 0.29
DAPO	26.67 ± 0.00	70.00 ± 0.00	84.88 ± 0.39	32.79 ± 0.66	55.05 ± 0.13	53.88 ± 0.16	60.73 ± 0.20
PrAg-PO (Step 1520)	33.33 ± 0.00	75.00 ± 0.00	85.12 ± 0.23	33.60 ± 0.56	53.87 ± 0.13	56.18 ± 0.12	60.69 ± 0.13
PrAg-PO (Step 720)	23.33 ± 0.00	76.50 ± 1.37	87.04 ± 0.43	33.53 ± 1.20	55.67 ± 0.44	55.22 ± 0.13	61.95 ± 0.25

Table 5: Evaluation statistics for Table 1 Qwen3-8B with standard deviation over the 5 inference rounds..

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	60.00 ± 0.00	95.00 ± 0.00	93.04 ± 0.09	46.10 ± 1.15	65.78 ± 0.38	71.98 ± 0.28	71.89 ± 0.33
DAPO	66.67 ± 0.00	90.00 ± 0.00	92.76 ± 0.26	44.78 ± 1.08	68.18 ± 0.53	72.48 ± 0.20	72.63 ± 0.24
PrAg-PO (Step 440)	70.00 ± 0.00	95.00 ± 0.00	94.28 ± 0.23	43.09 ± 0.16	69.27 ± 0.36	74.33 ± 0.08	73.51 ± 0.17
PrAg-PO (Step 380)	63.33 ± 0.00	87.50 ± 0.00	95.80 ± 0.55	45.59 ± 0.00	70.10 ± 0.28	72.47 ± 0.16	74.50 ± 0.29

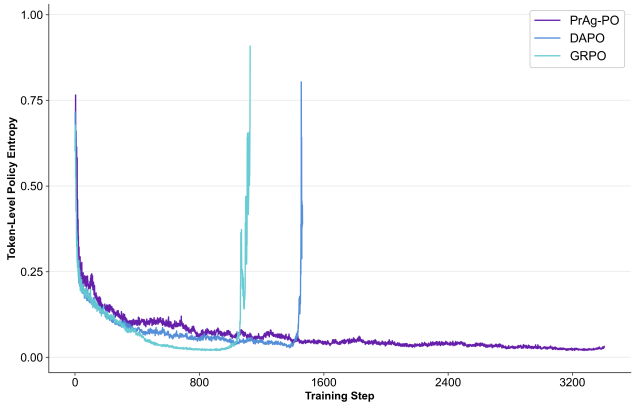

Figure 8: Token-level policy entropy during training of different methods on Qwen2.5-Math-1.5B.

A.2 Hyperparameters and Hardware Settings

For Qwen2.5-Math-1.5B and Qwen3-1.7B, we cap the maximum prompt length at 1024 tokens and the maximum output length at 3072 tokens during training, and use a maximum model length of 4096 tokens during evaluation. For DeepSeek-R1-Distill-Qwen-1.5B, following DRA-GRPO Chen et al. (2025b), we cap the maximum prompt length at 1024 tokens and the maximum output length at 3584 tokens during training, and use a maximum model length of 32768 tokens during evaluation. For Qwen3-8B, we cap the maximum prompt length at 1024 tokes and the maximum output length at 8192 tokens during training, and use a maximum model length of 9216 tokens during evaluation. AdamW optimizer and a constant learning rate of 1×10^{-6}. 1.5B and 1.7B training runs are conducted on 8 L40S GPUs, and 8B training runs are conducted on 4 B200 GPUs, with vLLM serving as the inference engine. Each training run takes approximately 1.5 - 4 days to complete depending on the total optimization steps.

A.3 Does Prompt Augmentation Increase Token-level Policy Entropy?

Given that prompt augmentation elicits diverse reasoning paths and output formats, a natural question is whether it also regularizes training by increasing token-level policy entropy. We find that the answer is nuanced: **As shown in Figure 8, Figure 9, and Figure 10, prompt augmentation helps stabilizing policy entropy instead of merely increasing it**.

For a large language model parameterized by θ, the policy π_θ defines a conditional distribution over the next token given the input question q and the previously generated tokens $o_{<t}$. The token-level policy entropy at decoding step t is defined as:

$$\mathcal{H}_t(\pi_\theta) = -\sum_{v \in \mathcal{V}} \pi_\theta(v \mid q, o_{<t}) \log \pi_\theta(v \mid q, o_{<t}),$$

where $\mathcal{V}$ denotes the vocabulary. In practice, token-level policy entropy aggregated over generated response tokens is computed as

$$\mathcal{H}_{\text{agg}}(\pi_\theta) = \frac{1}{\sum_{i=1}^{B} |o_i|} \sum_{i=1}^{B} \sum_{t=1}^{|o_i|} \mathcal{H}_{i,t}(\pi_\theta),$$

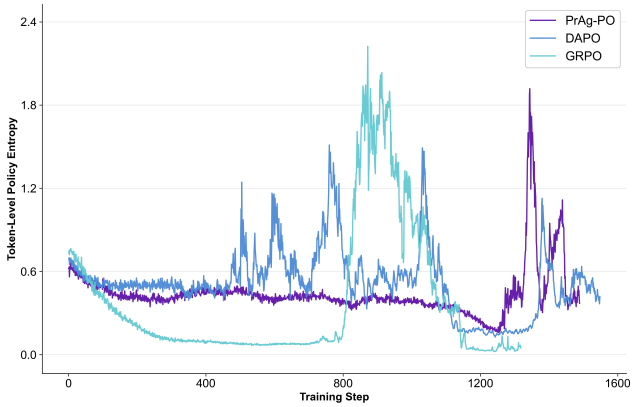

Figure 9: Token-level policy entropy during training of different methods on DeepSeek-R1-Distill-Qwen-1.5B

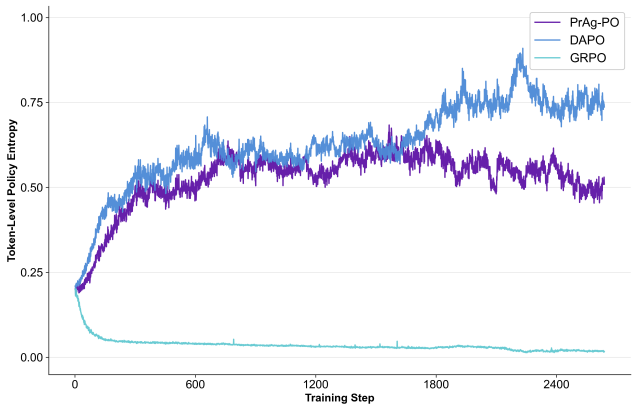

Figure 10: Token-level policy entropy during training of different methods on Qwen3-1.7B

where o_i denotes the i-th generated response in a batch B. Higher entropy corresponds to more stochastic and exploratory generation behavior. However, while token-level entropy measures **local uncertainty**, true diversity is inherently a sequence-level property. In principle, sequence-level entropy provides a more faithful characterization of this diversity, but it is intractable to compute for autoregressive models.

As illustrated in Figure 8, Figure 9, and Figure 10, training collapse is often accompanied by sharp entropy spikes and unstable fluctuations. While the token-level policy entropy of PrAg-PO (purple) is not consistently higher than that of DAPO, its entropy curve is notably smoother and exhibits a delayed onset of large fluctuations. This suggests that prompt augmentation stabilizes token-level policy entropy during training, rather than simply increasing it, while still promoting diversity through variation in reasoning paths and output formats.

A.4 Increasing KL Regularization Does Not Match PrAg-PO

Strong KL regularization constrains the model within narrower trust regions and has the potential to stabilize training, mitigate entropy collapse, and enable long-horizon optimization. Accordingly, we conduct GRPO training with a substantially larger KL coefficient ($\beta = 0.04$, as originally used in TRL) over an extended training horizon. As shown in Figure 11, although strong KL regularization effectively delays training collapse, it yields significantly lower performance than PrAg-PO. This is because strong KL constraints restrict exploratory behavior and keep the policy overly close to the base model. These results further highlight the effectiveness of PrAg-PO.

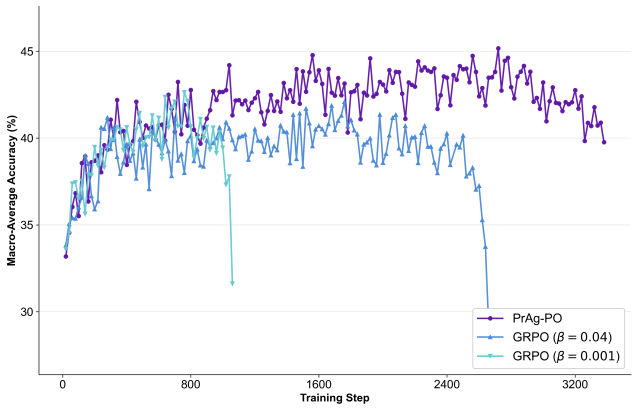

Figure 11: Evaluation accuracy over training steps with varying KL regularization coefficient. Increasing the KL coefficient in GRPO (darker blue) delays collapse but yields substantially lower accuracy.

A.5 Out-Of-Distribution Template Evaluation

To approximate an OOD setting, we use a minimal template that is relatively rare for instruction-tuned reasoning models: "{question}\nAnswer:" This template contains no system message, explicit instructions, or assistant-role prompt. We evaluate our trained checkpoints using this template, and the results are presented in Table 6, Table 7, and Table 8. PrAg-PO again outperforms both GRPO and DAPO on the majority of the benchmarks, achieving the highest average accuracies. These results are consistent with those in Table 1, which reports performance under the in-distribution templates. Notably, as shown in the first two tables below, DAPO exhibits a larger degradation under the OOD template, with its performance dropping below the GRPO baseline. In contrast, PrAg-PO consistently maintains its advantage. Overall, our findings suggest that PrAg-PO outperforms GRPO and DAPO under both in-distribution and out-of-distribution prompt styles, further demonstrating its robustness and generalizability.

Table 6: Evaluation results for Qwen2.5-MATH-1.5B using OOD template "{question}\nAnswer:".

Method	**AIME24**	**AMC(23&24)**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	**10.0**	21.7	40.4	10.3	**17.5**	20.0	23.7
DAPO	3.3	22.2	42.3	9.9	12.7	18.1	22.0
PrAg-PO (Step 2720)	**10.0**	**32.3**	**49.7**	**11.8**	15.6	**23.9**	**26.7**

Table 7: Evaluation results for DeepSeek-R1-Distill-Qwen-1.5B using OOD template "{question}\nAnswer:".

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	3.3	31.5	37.3	13.7	17.8	20.7	23.6
DAPO	6.7	10.0	26.2	10.7	11.7	13.0	16.2
PrAg-PO (Step 1100)	**16.7**	**41.0**	**42.5**	**16.1**	**25.3**	**28.3**	**29.6**

Table 8: Evaluation results for Qwen3-1.7B using OOD template "{question}\nAnswer:".

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olympiad**	**AVG** benchmark	**AVG** question
GRPO	**6.7**	11.5	33.8	**12.1**	12.3	15.3	19.2
DAPO	3.3	**25.0**	35.6	11.4	14.5	18.0	20.9
PrAg-PO (Step 1520)	**6.7**	**25.0**	**38.8**	10.7	**17.8**	**19.8**	**23.4**

B PROMPTS AND FORMAT REWARDS DETAILS

B.1 TEMPLATE STATISTICS VISUALIZATION

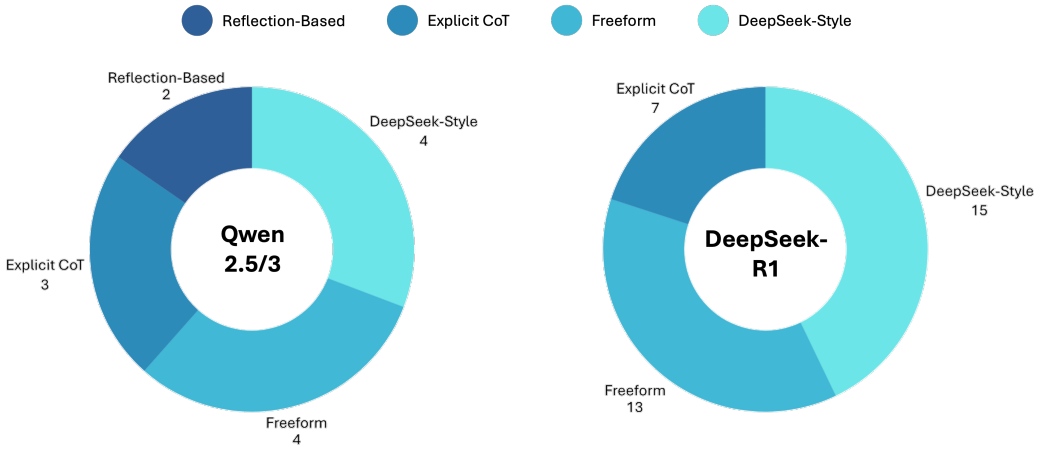

Figure 12: Template category distribution for Qwen2.5-Math-1.5B, Qwen3-1.7B, and Qwen3-8B (left), and DeepSeek-R1-Distill-Qwen-1.5B (right). Reflective templates are removed for DeepSeek-R1 because we observe that they slightly degrade training accuracy. We observe that DeepSeek-R1-Distill-Qwen-1.5B already exhibits frequent reflection and self-doubt behavior, and training with reflective templates can make this behavior excessive.

B.2 13 TEMPLATES FOR QWEN2.5 AND QWEN3

Table 9: 13 initial prompt templates used to train Qwen2.5-Math-1.5B, Qwen3-1.7B, and Qwen3-8B models, their associated categorizations, and format reward functions.

Prompt Template	**Category**	**Format Reward Function**

<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
Let's think step by step.
	Explicit CoT	`def format_reward(`
		`  completion):`
		`  return 1.0`

<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{question}
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>assistant
	Freeform	`def format_reward(`
		`  completion):`
		`  return 1.0`

<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
	Freeform	`def format_reward(`
		`  completion):`
		`  return 1.0`

<|im_start|>system
You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think>
...
</think>
<answer>
...
</answer>
Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n</think>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n<answer>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<|im_start|>system
You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think>
...
</think>
<answer>
...
</answer>
Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
</think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("\n</think>\n") == 1:`
		`  count += 1/3`
		`  if completion.count("\n<answer>\n") == 1:`
		`  count += 1/3`
		`  if completion.count("\n</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<|im_start|>system
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. The reasoning process is enclosed within <think> </think> and answer is enclosed within <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.25`
		`  if completion.count("</think>") == 1:`
		`  count += 0.25`
		`  if completion.count("<answer>") == 1:`
		`  count += 0.25`
		`  if completion.count("</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<|im_start|>system
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. The reasoning process is enclosed within <think> </think> and answer is enclosed within <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
<think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(`
		`  completion):`
		`  count = 0.0`
		`  if completion.count("</`
		`  think>") == 1:`
		`  count += 1/3`
		`  if completion.count("<`
		`  answer>") == 1:`
		`  count += 1/3`
		`  if completion.count("</`
		`  answer>") == 1:`
		`  count += 1/3`
		`  return count`

<|im_start|>system
You are an intelligent assistant who helps with user questions. Provide a rigorous, step-by-step derivation of the solution. The final answer must be clearly indicated within \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
	Freeform	`def format_reward(`
		`  completion):`
		`  return 1.0`

<|im_start|>system
Solve the following math challenge. Explain your approach step-by-step
The answer should end with: The final answer is: \boxed{answer}
where [answer] is just the final number or expression that solves the problem.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
Let's think step by step
	Explicit CoT	`def format_reward(`
		`  completion):`
		`  if completion.count("The`
		`  final answer is:")`
		`  == 1:`
		`  return 1.0`
		`  else:`
		`  return 0.0`

<|im_start|>system
Analyze and solve the math task.<|im_end|>
<|im_start|>user
{question}
End the answer with:
The final answer is: \boxed{answer}
where [answer] is just the final number or expression that solves the problem.<|im_end|>
<|im_start|>assistant
	Freeform	`def format_reward(`
		`  completion):`
		`  if completion.count("The`
		`  final answer is:")`
		`  == 1:`
		`  return 1.0`
		`  else:`
		`  return 0.0`

<|im_start|>system
Solve the following math problem
Show each step of your solution
Put the final answer within \boxed{answer}
where [answer] is just the final number or expression that solves the problem.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
Let's think step by step
	Explicit CoT	`def format_reward(`
		`  completion):`
		`  return  1.0`

<|im_start|>system
You are a helpful assistant that solves math problems. Always write out your reasoning to produce a solution, then check whether the solution is correct, fix it if it is wrong, and finally give the final answer. Respond in exactly the following format:
<solution>
reasoning and solution
</solution>
<check>
Let's verify step by step ...
</check>
Put your final answer within \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
	Reflection-Based	`def format_reward(`
		`  completion):`
		`  count = 0.0`
		`  if completion.count("<`
		`  solution>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n`
		`  </solution>\n") ==`
		`  1:`
		`  count += 0.25`
		`  if completion.count("\n<`
		`  check>\n Let's`
		`  verify step by step"`
		`  ) == 1:`
		`  count += 0.25`
		`  if completion.count("</`
		`  answer>") == 1:`
		`  count += 0.25`
		`  return  count`

<|im_start|>system
You are a helpful assistant that solves math problems. Always write out your reasoning to produce a solution, then check whether the solution is correct, fix it if it is wrong, and finally give the final answer. Respond in exactly the following format:
<solution>
reasoning and solution
</solution>
<check>
Let's verify step by step ...
</check>
Put your final answer within \boxed{}.<|im_end|>
<|im_start|>user
{question}<|im_end|>
<|im_start|>assistant
<solution>
	Reflection-Based (Teacher-Forced)	`def format_reward(`
		`  completion):`
		`  count = 0.0`
		`  if completion.count("\n`
		`  </solution>\n") ==`
		`  1:`
		`  count += 1/3`
		`  if completion.count("\n<`
		`  check>\n Let's`
		`  verify step by step"`
		`  ) == 1:`
		`  count += 1/3`
		`  if completion.count("\n`
		`  </check>") == 1:`
		`  count += 1/3`
		`  return  count`

B.3 35 TEMPLATES FOR DEEPSEEK-R1

Table 10: 35 prompt templates used to train the DeepSeek-R1-Distill-Qwen-1.5B model, their associated categorizations, and format reward functions. Note that DeepSeek-R1 use a different default template than Qwen, so we first map the initial Qwen templates to DeepSeek style (e.g., 'assistant' to '<|Assistant|>'). The first 11 templates come from the initial template set, and the next 23 templates are generated by a proprietary LLM. The last template is the default template of DeepSeek-R1-Distill-Qwen-1.5B. We removed reflective templates for this training because we found that they slightly degraded training accuracy. We observe that DeepSeek-R1-Distill-Qwen-1.5B already exhibits frequent reflection and self-doubt behavior, and training with reflective templates can make this behavior excessive. We add line breaks for presentation clarity; please refer to our codebase for the exact template strings.

Prompt Template	**Category**	**Format Reward Function**

<bos>Please reason step by step, and put your final answer within \boxed{}.
<|User|>{question}
<|Assistant|>Let's think step by step.
	Explicit CoT	`def format_reward(`
		`  completion):`
		`  return 1.0`

<bos>You are a helpful assistant.
<|User|>{question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|>
	Freeform	`def format_reward(`
		`  completion):`
		`  return 1.0`

<bos>Please reason step by step, and put your final answer within \boxed{}.
<|User|>{question}
<|Assistant|>
	Freeform	`def format_reward(`
		`  completion):`
		`  return 1.0`

<bos>You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think>
...
</think>
<answer>
...
</answer>
Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(`
		`  completion):`
		`  count = 0.0`
		`  if completion.count("<`
		`  think>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n`
		`  </think>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n<`
		`  answer>\n") == 1:`
		`  count += 0.25`
		`  if completion.count("\n`
		`  </answer>") == 1:`
		`  count += 0.25`
		`  return count`

<bos>You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think>
...
</think>
<answer>
...
</answer>
Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("\n</think>\n") == 1:`
		`  count += 1/3`
		`  if completion.count("\n<answer>\n") == 1:`
		`  count += 1/3`
		`  if completion.count("\n</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<bos>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. The reasoning process is enclosed within <think></think>and answer is enclosed within <answer></answer>tags, respectively, i.e., <think>reasoning process here </think><answer>answer here </answer>. Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.25`
		`  if completion.count("</think>") == 1:`
		`  count += 0.25`
		`  if completion.count("<answer>") == 1:`
		`  count += 0.25`
		`  if completion.count("</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<bos>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. The reasoning process is enclosed within <think></think>and answer is enclosed within <answer></answer>tags, respectively, i.e., <think>reasoning process here </think><answer>answer here </answer>. Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{}.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1/3`
		`  if completion.count("<answer>") == 1:`
		`  count += 1/3`
		`  if completion.count("</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<bos>You are an intelligent assistant who helps with user questions. Provide a rigorous, step-by-step derivation of the solution. The final answer must be clearly indicated within \boxed{}. <|User|>{question} <|Assistant|>	Freeform	`def format_reward(completion):` `  return 1.0`

<bos>Solve the following math challenge. Explain your approach step-by-step
The answer should end with: The final answer is: \boxed{answer}
where [answer] is just the final number or expression that solves the problem
<|User|>{question}
<|Assistant|>Let's think step by step
	Explicit CoT	`def format_reward(completion):`
		`  key = "The final answer is:"`
		`  format_score = 1.0 if completion.count(key) == 1 else 0.0`
		`  return format_score`

<bos>Analyze and solve the math task.
<|User|>{question}
End the answer with:
The final answer is: \boxed{answer}
where [answer] is just the final number or expression that solves the problem.
<|Assistant|>
	Freeform	`def format_reward(completion):`
		`  key = "The final answer is:"`
		`  format_score = 1.0 if completion.count(key) == 1 else 0.0`
		`  return format_score`

<bos>Solve the following math problem Show each step of your solution Put the final answer within \boxed{} <|User|>{question} <|Assistant|>Let's think step by step	Explicit CoT	`def format_reward(completion):` `  return 1.0`
<bos>Solve step by step and ensure the final answer is enclosed in \boxed{}. <|User|>{question} <|Assistant|>Let's solve this step by step.	Explicit CoT	`def format_reward(completion):` `  return 1.0`
<bos>Carefully derive the solution step by step. Final answer must be in \boxed{}. <|User|>{question} <|Assistant|>	Freeform	`def format_reward(completion):` `  return 1.0`
<bos>Provide a detailed reasoning process. Conclude with \boxed{}. <|User|>{question} <|Assistant|>Start reasoning:	Explicit CoT	`def format_reward(completion):` `  return 1.0`
<bos>Break the problem into steps and solve sequentially. Final answer in \boxed{}. <|User|>{question} <|Assistant|>	Freeform	`def format_reward(completion):` `  return 1.0`
<bos>Follow strict reasoning. Do not skip steps. Final answer must be in \boxed{}. <|User|>{question} <|Assistant|>	Freeform	`def format_reward(completion):` `  return 1.0`
<bos>Explain clearly and end precisely with \boxed{}. <|User|>{question} <|Assistant|>Proceed carefully:	Explicit CoT	`def format_reward(completion):` `  return 1.0`
<bos>Ensure logical correctness at every step. Final answer strictly inside \boxed{}. <|User|>{question} <|Assistant|>	Freeform	`def format_reward(completion):` `  return 1.0`

<bos>Solve the problem and show reasoning. End with:
We conclude that the answer is: \boxed{answer}
<|User|>{question}
<|Assistant|>
	Freeform	`def format_reward(completion):`
		`  key = "We conclude that the answer is:"`
		`  return 1.0 if completion.count(key) == 1 else 0.0`

<bos>Work through the math problem step by step. Finish with:
Therefore, the final answer is \boxed{answer}
<|User|>{question}
<|Assistant|>Let's think carefully.
	Explicit CoT	`def format_reward(completion):`
		`  key = "Therefore, the final answer is"`
		`  return 1.0 if completion.count(key) == 1 else 0.0`

<bos>Provide reasoning and ensure the answer ends with:
Based on the analysis, the answer is: \boxed{answer}
<|User|>{question}
<|Assistant|>
	Freeform	`def format_reward(completion):`
		`  key = "Based on the analysis, the answer is:"`
		`  return 1.0 if completion.count(key) == 1 else 0.0`

<bos>Explain your reasoning step by step, and conclude with a final statement of the form:
From the steps above, the answer is \boxed{answer}
<|User|>{question}
<|Assistant|>
	Freeform	`def format_reward(completion):`
		`  key = "From the steps above, the answer is "`
		`  return 1.0 if completion.count(key) == 1 else 0.0`

<bos>Provide concise but sufficient reasoning. Final answer must be in \boxed{}. <|User|>{question} <|Assistant|>	(Concise) Freeform	`def format_reward(completion):` `  return 1.0`
<bos>Solve briefly but include key steps. End with \boxed{}. <|User|>{question} <|Assistant|>	(Concise) Freeform	`def format_reward(completion):` `  return 1.0`

<bos>Solve the following problem.
Format:
<think>
...
</think>
<answer>
\boxed{}
</answer>
Ensure correctness and proper formatting.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("n</think>n") == 1:`
		`  count += 1/3`
		`  if completion.count("n<answer>n") == 1:`
		`  count += 1/3`
		`  if completion.count("n</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<bos>Solve the following problem.
Format:
<think>
...
</think>
<answer>
\boxed{}
</answer>
Ensure correctness and proper formatting.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>n") == 1:`
		`  count += 0.25`
		`  if completion.count("n</think>n") == 1:`
		`  count += 0.25`
		`  if completion.count("n<answer>n") == 1:`
		`  count += 0.25`
		`  if completion.count("n</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<bos>Solve the problem carefully and ensure correctness.
Use this exact format:
<think>...</think><answer>...</answer>
The answer must be enclosed in \boxed{}.
All tags must be properly closed.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1/3`
		`  if completion.count("<answer>") == 1:`
		`  count += 1/3`
		`  if completion.count("</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<bos>Solve the problem carefully and ensure correctness.
Use this exact format:
<think>...</think><answer>...</answer>
The answer must be enclosed in \boxed{}.
All tags must be properly closed.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.25`
		`  if completion.count("</think>") == 1:`
		`  count += 0.25`
		`  if completion.count("<answer>") == 1:`
		`  count += 0.25`
		`  if completion.count("</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<bos>You are given a problem to solve.
Use this structure exactly:
<think>
...
</think>
<answer>
...
</answer>
Final answer must be enclosed in \boxed{}.
Do not skip steps.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1/3`
		`  if completion.count("<answer>") == 1:`
		`  count += 1/3`
		`  if completion.count("</answer>") == 1:`
		`  count += 1/3`
		`  return count`

<bos>You are given a problem to solve.
Use this structure exactly:
<think>
...
</think>
<answer>
...
</answer>
Final answer must be enclosed in \boxed{}.
Do not skip steps.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.25`
		`  if completion.count("</think>") == 1:`
		`  count += 0.25`
		`  if completion.count("<answer>") == 1:`
		`  count += 0.25`
		`  if completion.count("</answer>") == 1:`
		`  count += 0.25`
		`  return count`

<bos>Your task is to solve the given problem correctly.
All reasoning must be inside <think></think>.
The final answer must be enclosed in \boxed{}.
Ensure correctness before answering.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1`
		`  return count`

<bos>Your task is to solve the given problem correctly.
All reasoning must be inside <think></think>.
The final answer must be enclosed in \boxed{}.
Ensure correctness before answering.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.5`
		`  if completion.count("</think>") == 1:`
		`  count += 0.5`
		`  return count`

<bos>Solve the problem step by step.
Organize your reasoning clearly inside <think></think>.
The final answer must be in \boxed{}.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-Style (Teacher-Forced)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1`
		`  return count`

<bos>Solve the problem step by step.
Organize your reasoning clearly inside <think></think>.
The final answer must be in \boxed{}.
<|User|>{question}
<|Assistant|>
	DeepSeek-Style	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("<think>") == 1:`
		`  count += 0.5`
		`  if completion.count("</think>") == 1:`
		`  count += 0.5`
		`  return count`

<bos>Please reason step by step, and put your final answer within \boxed{}.
<|User|>{question}
<|Assistant|><think>
	DeepSeek-style (Default Template)	`def format_reward(completion):`
		`  count = 0.0`
		`  if completion.count("</think>") == 1:`
		`  count += 1`
		`  return count`

B.4 Format Reward Best Practices

Empirically, we find that simple additive format rewards often achieve the best performance. In contrast, more complex conditional logic or regular-expression-based rewards can degrade reasoning accuracy. In particular, rules that overly restrict the model's behavior may accelerate training collapse and lead to worse outcomes. For example, the format reward in Listing 49 limits the number of characters generated after the model's final boxed answer to fewer than 70, with the goal of preventing the model from continuing to output gibberish after completing its answer. However, we find that this reward structure can induce early collapse and substantially degrade performance.

```python
def find_matching_brace(s, start):
    depth = 1 # assumes s[start] == "{"
    for i in range(start + 1, len(s)):
        c = s[i]
        if c == "{":
            depth += 1
        elif c == "}":
            depth -= 1
            if depth == 0:
                return i
    return -1

def boxed_format_and_length_penalty(completion):
    key = r"\boxed"
    idx = completion.rfind(key)
    if idx == -1:
        return 0.0

    # find opening brace after \boxed
    brace_start = completion.find("{", idx)
    if brace_start == -1:
        return 0.0

    # find matching closing brace
    brace_end = find_matching_brace(completion, brace_start)
    if brace_end == -1:
        return 0.0

    # count chars after boxed expression
    tail_len = len(completion) - (brace_end + 1)

    if tail_len > 70:
        return 0.0

    return 1.0

# should be very limited words after last boxed
def abel_format_reward(completion):
    return boxed_format_and_length_penalty(completion)
```

Listing 49: abel_format_reward is an example of an overly complex and restrictive format reward function, which requires the model to generate fewer than 70 characters after its final answer.