CPGD: Toward Stable Rule-based Reinforcement Learning for Language Models

Zongkai Liu[1,2]* **Fanqing Meng**[4]* **Lingxiao Du**[3]* **Zhixiang Zhou**[2]*
Chao Yu[1]† **Wenqi Shao**[2,3]† **Qiaosheng Zhang**[2,3]†

[1]**Sun Yat-Sen University** [2]**Shanghai Innovation Institute**
[3]**Shanghai AI Laboratory** [4]**Shanghai Jiao Tong University**

Abstract

Recent advances in rule-based reinforcement learning (RL) have significantly improved the reasoning capability of language models (LMs) with rule-based rewards. However, existing RL methods—such as GRPO, REINFORCE++, and RLOO—often suffer from training instability, where large policy updates and improper clipping can lead to training collapse. To address this issue, we propose Clipped Policy Gradient Optimization with Policy Drift (CPGD), a novel algorithm designed to stabilize policy learning in LMs. CPGD introduces a policy drift constraint based on KL divergence to dynamically regularize policy updates, and leverages a clip mechanism on the logarithm of the ratio to prevent excessive policy updates. We provide theoretical justification for CPGD and demonstrate through empirical analysis that it mitigates the instability observed in prior approaches. Furthermore, we show that CPGD significantly improves performance while maintaining training stability. Our implementation balances theoretical rigor with practical usability, offering a robust alternative for RL in the post-training of LMs. We release our code at https://github.com/ModalMinds/MM-EUREKA.

1 Introduction

Rule-based reinforcement learning (RL) has emerged as a key approach for eliciting reasoning capabilities in language models (LMs) [1]. It leverages simple, efficient reward functions derived from deterministic rules, effectively mitigating reward hacking [2] while activating reasoning abilities of models [1, 3, 4, 5]. This has sparked a line of research focused on developing more effective RL algorithms for both textual and general multimodal reasoning tasks. Notable methods include GRPO [1], REINFORCE++ [6], RLOO [7, 8], and GRPO variants such as DAPO [9], Dr.GRPO [10], and GPG [11]. However, we observe that these RL methods often suffer from training instability, which we attribute to the use of *importance-sampling ratios* in their loss functions. Although PPO-clip loss [12] is commonly adopted to mitigate extreme policy updates, its one-sided nature fails to constrain large ratios when the advantage is negative—potentially causing gradient explosions dominated by poor samples, leading to catastrophic training collapse. We theoretically show that incorporating the importance-sampling ratio in the loss can amplify the policy shift, and our empirical results confirm that this can lead to training collapse in existing RL methods.

To address this issue, we propose *Clipped Policy Gradient Optimization with Policy Drift* (CPGD), an algorithm that replaces the PPO-clip loss with a policy gradient loss [13] to avoid instability caused by directly involving policy ratios in the loss function. To ensure proximal optimization, we introduce both a clip mechanism and a policy drift regularizer, constraining optimization within a local region

*Equal contribution

†Corresponding Authors: {zhangqiaosheng, shaowenqi}@pjlab.org.cn; yuchao3@mail.sysu.edu.cn

Preprint. Under review.

and mitigating over-optimization that may impair reasoning behaviors as shown in Section 4.2. Furthermore, we develop a novel KL estimator that ensures correct gradient directions while avoiding the potential numerical instability associated with the commonly used k_3 estimators [14]. We also incorporate weighted advantages to dynamically adjust the influence of each sample, further enhancing model performance.

We theoretically prove the convergence of CPGD and empirically demonstrate its superior training stability and performance. As shown in Table 1, models trained with CPGD consistently outperform those trained with other RL algorithms and strong open-source baselines across standard multimodal reasoning benchmarks. Notably, CPGD improves the overall performance over the base model by +11.0% across all benchmarks. Specially, CPGD achieves +21.8% gain on the in-domain benchmark MMK12 [15], and improves by +8.5% and +11.4% on the out-of-distribution benchmarks MathVista [16] and MathVision [17], respectively.

2 Related work

RL for training reasoning models. RL has become a key method for improving reasoning in LMs [1, 18]. While early methods rely on PPO [12], its high computational cost has driven interest in alternatives like DPO [19], which simplifies training but depends on offline data. Recent RL methods such as GRPO, RLOO, and REINFORCE++ aim to balance stability and efficiency. Notably, DeepSeek R1 [1] shows that pure RL can elicit self-reflection and reasoning in LMs without supervised pretraining. Recently, several concurrent works have proposed GRPO variants to address its limitations. For instance, Dr.GRPO [10] identifies optimization bias in GRPO that favors longer response among incorrect ones. DAPO [9] incorporates multiple improvements, including decoupled clipping thresholds, token-level losses, and an online filtering strategy. GPG [11], in contrast, adopts a minimalist design by discarding both clipping and KL regularization, relying solely on the policy gradient loss [13]. However, none of these approaches fundamentally resolve the training instability issue to existing RL methods, which is the primary focus of this work.

Large reasoning model. Recently, a surge of reasoning models has emerged, driven by the principle of test-time scaling laws, which demonstrate that models with explicit reasoning processes achieve superior performance [20]. Leading models in this area include DeepSeek R1 [1], OpenAI's o-series [18], Qwen series [21, 22], and Kimi k1.5 [23]. However, their training pipelines and datasets remain undisclosed. This has motivated a wave of academic research within the open-source community, including parallel efforts such as OpenR1 [24], TinyZero [25], LMM-R1 [26], R1-V [27], Reason-RFT [28], and MM-Eureka [15]. These works primarily focus on constructing high-quality datasets and complete training pipelines. They commonly adopt GRPO to enhance reasoning capabilities but do not specifically investigate improvements to the RL algorithms themselves.

3 Preliminaries

3.1 Problem formulation

We denote an LM by π_θ, where $\theta \in \mathbb{R}^d$ represents the model parameters. Given a prompt $\mathbf{x} = [x_1, \ldots, x_m] \in \mathcal{D}$, the model generates a response $\mathbf{y} = [y_1, \ldots, y_n]$ by sampling from the conditional distribution $\pi_\theta(\cdot|\mathbf{x})$, with both x_i and y_i drawn from a finite vocabulary $\mathcal{V}$. In this work, we focus on transformer-based LMs that generate responses autoregressively, such that $\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} \pi_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i})$, where $\mathbf{y}_{<i} = [y_1, \ldots, y_{i-1}]$ and $\mathbf{y}_{<1}$ is an empty sequence.

RL in post-training is typically modeled as a Markov decision process (MDP), defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition kernel, $\mathcal{R}$ is the deterministic reward function, and ρ is the initial state distribution. For LMs, two MDP formulations are widely considered: *token-level MDP* and *response-level MDP*.

In a *token-level MDP*, each token is treated as a single action. At the time step t, the state $\mathbf{s}_t = [\mathbf{x}, \mathbf{y}_{<t}]$ includes the prompt and the tokens generated so far. The action $a_t = y_t$ is sampled according to $y_t \sim \pi_\theta(\cdot|\mathbf{x}, \mathbf{y}_{<t})$, where the action space $\mathcal{A}$ is equal to the vocabulary $\mathcal{V}$. The environment transitions deterministically to $\mathbf{s}_{t+1} = [\mathbf{x}, \mathbf{y}_{<t+1}]$. The reward is defined as $\mathcal{R}(\mathbf{s}_t, a_t) = \mathcal{R}([\mathbf{x}, \mathbf{y}_{<t}], y_t)$, and ρ is induced by the prompt distribution in $\mathcal{D}$.

In a *response-level MDP*, the full response is treated as an individual action: $\mathbf{a} = \mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})$. The state is defined solely by the prompt $\mathbf{s} = \mathbf{x}$, and the episode terminates after one step. Thus, the transition kernel is omitted in the single-turn dialogue setting. The reward is $\mathcal{R}(\mathbf{s}, \mathbf{a}) = \mathcal{R}(\mathbf{x}, \mathbf{y})$, with ρ again determined by $\mathcal{D}$.

3.2 Rule-based reinforcement learning

This work focuses on verifiable tasks, where the outcome reward is determined by the final accuracy. Specifically, a response $\mathbf{y}$ receives a reward of 1 if it is the correct answer to the prompt $\mathbf{x}$, and 0 otherwise. We denote this reward function as $\mathcal{R}_o$ to emphasize its nature as an outcome-based reward. Within this setting, REINFORCE-style algorithms are favored as they reduce computational cost by forgoing critic networks. Notable methods include REINFORCE++ [6], RLOO [7, 8], and GRPO [1].

REINFORCE++: REINFORCE++ enhances the standard REINFORCE framework by integrating key optimizations from PPO [12], improving both stability and efficiency. The objective is defined as:

$$\mathcal{L}_{\text{R++}}(\theta; \theta_{old}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}, \mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})} \left[\frac{1}{|\mathbf{y}|} \sum_{i=1}^{|\mathbf{y}|} \min \left(\frac{\pi_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x}, \mathbf{y}_{<i})} A_i^{\text{R++}}, \right. \right.$$
$$\left. \left. \text{clip}_{1-\epsilon}^{1+\epsilon} \left(\frac{\pi_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x}, \mathbf{y}_{<i})} \right) A_i^{\text{R++}} \right) \right],$$

where $\epsilon \in [0, 1]$, $\text{clip}_a^b(x) := \max(\min(x, b), a)$, and

$$A_i^{\text{R++}} := \text{GlobalNorm}\left(G(\mathbf{x}, \mathbf{y}_{\le i})\right), \quad G(\mathbf{x}, \mathbf{y}_{\le i}) := \mathcal{R}_o(\mathbf{x}, \mathbf{y}) - \beta \sum_{j=i}^{|\mathbf{y}|} \ln \frac{\pi_{\theta_{old}}(y_j|\mathbf{x}, \mathbf{y}_{<j})}{\pi_{\text{ref}}(y_j|\mathbf{x}, \mathbf{y}_{<j})}.$$

Here, $\ln \frac{\pi_{\theta_{old}}(y_j|\mathbf{x}, \mathbf{y}_{<j})}{\pi_{\text{ref}}(y_j|\mathbf{x}, \mathbf{y}_{<j})}$ is the token-level KL penalty, constraining divergence from the reference policy π_{ref}, typically the initial model. $\text{GlobalNorm}(x) = \frac{x - \text{mean}(\{x' \in \text{batch}\})}{\text{std}(\{x' \in \text{batch}\})}$ is the normalization operation across the global batch for all prompts.

RLOO: The primary distinction between RLOO and REINFORCE++ lies in their computation of the advantage value. RLOO first generates a group of K responses $\{\mathbf{y}^{(k)}\}_{k=1}^K$ for each prompt $\mathbf{x}$ and computes the advantage using a *leave-one-out* strategy to reduce the gradient variance:

$$A_{i,k}^{\text{RLOO}} := \text{GlobalNorm}\left(\tilde{G}(\mathbf{x}, \mathbf{y}_{\le i}^{(k)})\right), \quad \tilde{G}(\mathbf{x}, \mathbf{y}_{\le i}^{(k)}) := G(\mathbf{x}, \mathbf{y}_{\le i}^{(k)}) - \frac{1}{K-1} \sum_{k' \ne k} G(\mathbf{x}, \mathbf{y}_{\le i}^{(k')}).$$

GRPO: GRPO introduces a group-based advantage and employs an external KL regularization via the k_3 estimator [14], which approximates $D_{\text{KL}}(p, q) = \sum_i (q_i/p_i - 1 - \ln q_i/p_i)$. The loss is:

$$\mathcal{L}_{\text{GRPO}}(\theta; \theta_{old}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}, \{\mathbf{y}^{(k)}\}_{k=1}^K \sim \pi_{\theta_{old}}(\cdot|\mathbf{x})} \left[\frac{1}{K} \sum_{k=1}^K \left(\frac{1}{|\mathbf{y}^{(k)}|} \sum_{i=1}^{|\mathbf{y}^{(k)}|} \left(-\beta \cdot \mathcal{M}_{\theta,\text{ref}}^i(\mathbf{x}, \mathbf{y}^{(k)}) \right. \right. \right.$$
$$\left. \left. \left. + \min \left(\frac{\pi_\theta(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})}{\pi_{\theta_{old}}(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})} A_k^{\text{GRPO}}, \text{clip}_{1-\epsilon}^{1+\epsilon} \left(\frac{\pi_\theta(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})}{\pi_{\theta_{old}}(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})} \right) A_k^{\text{GRPO}} \right) \right) \right],$$

where

$$A_k^{\text{GRPO}} := \text{GroupNorm}(\mathcal{R}_o(\mathbf{x}, \mathbf{y}^{(k)})) = \frac{\mathcal{R}_o(\mathbf{x}, \mathbf{y}^{(k)}) - \text{mean}(\{\mathcal{R}_o(\mathbf{x}, \mathbf{y}^{(k)})\}_{k=1}^K)}{\text{std}(\{\mathcal{R}_o(\mathbf{x}, \mathbf{y}^{(k)})\}_{k=1}^K)},$$

$$\mathcal{M}_{\theta,\text{ref}}^i(\mathbf{x}, \mathbf{y}^{(k)}) := \frac{\pi_{\text{ref}}(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})}{\pi_\theta(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})} - 1 - \ln \frac{\pi_{\text{ref}}(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})}{\pi_\theta(y_i^{(k)}|\mathbf{x}, \mathbf{y}_{<i}^{(k)})}.$$

4 The proposed method

This section introduces our RL algorithm, *Clipped Policy Gradient Optimization with Policy Drift* (CPGD), designed to improve the stability of RL training. In Section 4.1, we present the CPGD

algorithm along with its theoretical guarantees, and highlight potential limitations of the standard PPO-clip loss. In Section 4.2, we provide empirical evidence of instability in existing methods and analyze its possible causes, showing how CPGD addresses them for more stable training. Finally, Section 4.3 describes the practical implementation of CPGD, striking a balance between theoretical soundness and practical implementation.

4.1 Clipped Policy Gradient Optimization with Policy Drift (CPGD)

Under the response-level MDP assumption, CPGD aims to maximize the following formula:

$$\mathcal{L}_{\text{CPGD}}(\theta;\theta_{old}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})}\big[\Phi_\theta(\mathbf{x},\mathbf{y})\big] - \alpha\cdot D_{\text{KL}}(\pi_{\theta_{old}},\pi_\theta|\mathbf{x})\Big], \tag{1}$$

where

$$\Phi_\theta(\mathbf{x},\mathbf{y}) := \min\Big(\ln\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}\cdot A^{\text{CPGD}}(\mathbf{x},\mathbf{y}), \text{clip}_{\ln(1-\epsilon)}^{\ln(1-\epsilon)}\Big(\ln\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}\Big)A^{\text{CPGD}}(\mathbf{x},\mathbf{y})\Big),$$

$$A^{\text{CPGD}}(\mathbf{x},\mathbf{y}) := \mathcal{R}_o(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}\big[\mathcal{R}_o(\mathbf{x},\mathbf{y}')\big],$$

$$D_{\text{KL}}(\pi_{\tilde{\theta}},\pi_\theta|\mathbf{x}) := \mathbb{E}_{\mathbf{y}\sim\pi_{\tilde{\theta}}(\cdot|\mathbf{x})}\Big[\ln\frac{\pi_{\tilde{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_\theta(\mathbf{y}|\mathbf{x})}\Big].$$

Hereinafter, we term the KL divergence between the old and current policies as *policy drift*, and between the current and reference policies as *reference constraint*.

CPGD differs from the standard PPO-clip loss in two key aspects: (1) A different policy optimization objective is used, where the policy gradient loss is adopted with the clip mechanism. (2) A policy drift is introduced, imposing a forward KL divergence penalty between the old and current policies.

Why use the policy gradient objective? In the original PPO objective, although the importance-sampling ratio corrects for the distribution mismatch between the old and current policies, it simultaneously introduces high variance. As empirically demonstrated in Section 4.2, such variance can destabilize training and even cause training collapse, while using a policy gradient loss without the ratio substantially improves training stability. Proposition 1 further provides a theoretical explanation for this phenomenon, showing that the use of the policy ratio amplifies policy drift, causing the updated policy to exceed the intended bounds.

Why introduce the policy drift and clip mechanism? The introduction of the clip mechanism and policy drift is designed to ensure proximal policy updates, which are critical for theoretical convergence guarantees in Theorem 1. The clip mechanism enforces local updates by zeroing gradients when the policy ratio exceeds a specified threshold, while policy drift introduces a corrective gradient to constrain the policy ratio within a stable range. Notably, the clip mechanism alleviates the need for a large penalty coefficient on the policy drift term: when the ratio remains within bounds, the small drift coefficient allows the algorithm to focus on optimizing the primary objective Φ; when the ratio exceeds the range, the gradient of the primary objective becomes zero, prompting the algorithm to rely on the policy drift signal to prevent further deviation—particularly those caused by optimizer momentum (e.g., Adam) or neural network generalization effects.

Proposition 1. *Let θ_0 be a parameter such that the importance-sampling ratio satisfies $|\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1| = \epsilon$. Consider updating θ_0 using either (i) the PPO-clip objective, resulting in parameter θ_1^{PPO}, or (ii) the CPGD objective with $\alpha = 0$ (denoted as CPG), yielding parameter θ_1^{CPG}. Then, there exists a constant $\eta_{\max} > 0$ such that for any learning rate $\eta \in (0, \eta_{\max})$, the following inequality holds:*

$$\left|\frac{\pi_{\theta_1^{PPO}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| > \left|\frac{\pi_{\theta_1^{CPG}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| > \left|\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| = \epsilon.$$

After one update step, both PPO and CPG increase the importance-sampling ratio deviation from the old policy, but PPO does so more aggressively than CPG.

The following theorem further presents that CPGD enjoys the convergence guarantee, indicating its theoretical rationality. See Appendix A for the proofs of Proposition 1 and Theorem 1.

Theorem 1. *Let $\{\pi_{\theta_k}\}_{k=0}^{\infty}$ denote the sequence of policies generated by the CPGD update rule (Equation 1). Then, the sequence π_{θ_k} converges.*

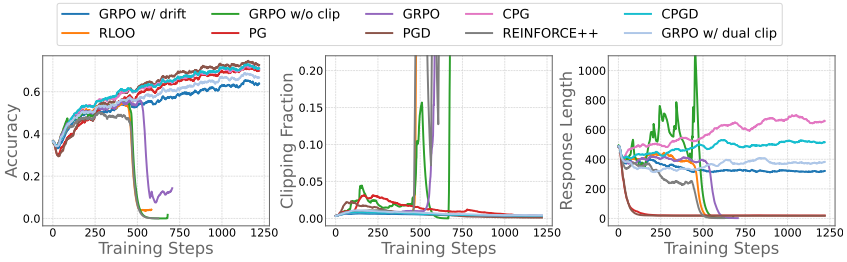

Figure 1: Accuracy, clipping fraction and response length curves throughout training.

4.2 Training collapse

Several studies suggest that the reference constraint may hinder policy improvement [9, 29]. However, we observe that removing this KL term leaves the PPO-clip loss alone insufficient to effectively constrain large policy shifts, which can lead to training collapse. While such collapse may be partially mitigated through techniques such as early stopping or small learning rates, it remains a latent instability that undermines the reliability of continued training. In this subsection, we investigate this phenomenon of training collapse and demonstrate that CPGD effectively prevents it.

Figure 1 presents training curves on the MMK12 dataset [15] for RLOO, REINFORCE++, GRPO, GRPO w/o clip (i.e., GRPO without the clip mechanism), GRPO w/ dual clip (i.e., the policy ratio is additionally clipped to no more than a constant—3.0 in our case—when advantage is negative [30]), GRPO w/ drift (i.e., GRPO with policy drift), PG (basic policy gradient), CPG (PG with the clip mechanism), PGD (PG with the policy drift), and CPGD, all without the reference constraint. We use QwenVL2.5-7B [31] as the base model. All algorithms share the same hyperparameters: a training and rollout batch size of 128, 8 responses per prompt, a learning rate of $1e{-}6$, one PPO epoch, and ten training episodes. As shown in Figure 1, almost all baselines experience training collapse.

As shown in Figure 1, methods such as REINFORCE++, RLOO, GRPO w/o clip, and GRPO exhibit highly unstable policy ratio dynamics, leading to training collapse in mid stages. In contrast, GRPO w/ dual clip, GRPO w/ drift, PG, CPG, PGD, and CPGD maintain stable training curves. GRPO w/ dual clip mitigates instability by globally constraining the policy ratio, while the PG series sidesteps ratio-induced variance by excluding it from the loss computation. These comparisons indicate that incorporating policy ratios in the loss can introduce high variance during fluctuations, and that simple one-sided clipping fails to recover from extreme ratios, ultimately causing collapse. Although dual clip mechanism stabilizes training, it may introduce new issues: frequent zero-gradient updates and ineffective learning under negative advantages due to the zero-gradient clipped large ratios. Additionally, GRPO w/ drift demonstrates that incorporating policy drift effectively constrains the policy ratio within a reasonable range, thereby preventing training collapse.

On the other hand, while prior work suggests clipping may be unnecessary due to the low proportion of clipped ratios [8, 11], our findings suggest otherwise. Despite only ~1% of ratios being clipped, training performance diverges significantly with and without clipping. Specifically, methods like PG and PGD—though stable without ratio terms—suffer from response length collapse, degenerating into trivial outputs (e.g., only emitting tokens like <think>) that exploit the format reward function without performing meaningful reasoning. This highlights the model's vulnerability to reward hacking, likely due to overly aggressive updates. These results reveal the necessity of the proximal policy updates.

4.3 Implementation

In this subsection, we design a practically implementable loss function in per-token form based on the CPGD update formulation (Equation 1), aiming to strike a balance between theoretical rigor and empirical applicability. Our CPGD loss is straightforward to be integrated into widely-used large model training frameworks such as OpenRLHF [32] and veRL [33]. The practical loss function is given by

$$\mathcal{J}_{\text{CPGD}}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\{\mathbf{y}^{(k)}\}_{k=1}^{K})\in\mathcal{D}} \frac{1}{\sum_{k=1}^{K}|\mathbf{y}^{(k)}|} \left[\sum_{i=1}^{|\mathbf{y}^{(k)}|} \left(\Phi_\theta^i(\mathbf{x},\mathbf{y}^{(k)}) - \alpha \cdot \mathcal{E}^i_{\theta_{old},\theta}(\mathbf{x},\mathbf{y}^{(k)}) \right) \right], \quad (2)$$

where the per-token policy optimization term is

$$\Phi_\theta^i(\mathbf{x},\mathbf{y}) := \min\left(\ln\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}\cdot A_\omega^{\text{CPGD}}(\mathbf{x},\mathbf{y}),\ \text{clip}_{\ln(1-\epsilon_i)}^{\ln(1+\epsilon_i)}\Big(\ln\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}\Big)A_\omega^{\text{CPGD}}(\mathbf{x},\mathbf{y})\right),$$

and

$$A_\omega^{\text{CPGD}}(\mathbf{x},\mathbf{y}^{(k)}) := \omega(\mathbf{x})\cdot\Big(\mathcal{R}_o(\mathbf{x},\mathbf{y}^{(k)}) - \text{mean}\left(\{\mathcal{R}_o(\mathbf{x},\mathbf{y}^{(k')})\}_{k'=1}^K\right)\Big),$$

$$\mathcal{E}^i_{\theta_{old},\theta}(\mathbf{x},\mathbf{y}) := \min\left(\frac{\text{sg}(\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}))}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})} - 1, c\right)\cdot\ln\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}).$$

Here, $\text{sg}(\cdot)$ denotes the operation that prevents gradient computation, $\omega(\mathbf{x})$ is a per-prompt weighting factor, and $c > 0$ is a constant. We provide the following clarifications regarding the differences between the theoretical update formulation (Equation 1) and the practical loss (Equation 2):

(I) Policy optimization term: In the theoretical update (Equation 1), the policy optimization term is written in the form of joint distribution. But in the practical implementation (Equation 2), it is decomposed into token level using the decomposability of the logarithm function. Specifically, the clipping threshold ϵ_i can be set the same for all tokens, ensuring that each token shares the same clip range. Alternatively, a tight-to-loose schedule can be employed such as $\epsilon_i = \lambda\epsilon + (1-\lambda)\epsilon\cdot i/|\mathbf{y}^{(k)}|$, which assigns smaller thresholds to earlier tokens that usually have higher variance.

(II) Policy drift: Similar to the policy optimization term, policy drift also leverages the decomposability of the logarithm function, but applies the following further transformations:

$$D_{\text{KL}}(\pi_{\theta_{old}},\pi_\theta|\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})}\Big[\ln\frac{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}{\pi_\theta(\mathbf{y}|\mathbf{x})}\Big] = \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})}\Big[\sum_{i=1}^{|\mathbf{y}|}\ln\frac{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}\Big] \tag{3}$$

$$= \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})}\Big[\sum_{i=1}^{|\mathbf{y}|}\Big(\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})} - 1 - \ln\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}\Big)\Big]. \tag{4}$$

Equations 3 and 4 correspond to the k_1 and k_3 KL estimators proposed by Schulman [14]. In practice, particularly when using gradient optimizers such as Adam, we prefer the k_3 estimator over k_1, as k_1 fails to effectively constrain the policy drift, while the gradient direction of k_3 dynamically adjusts based on the relative magnitude between the current and old policies:

$$\begin{aligned}
&\nabla_\theta\ln\frac{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})} = -\nabla_\theta\ln\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}),\\
&\nabla_\theta\Big(\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})} - 1 - \ln\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})}\Big) = \Big(\frac{\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})} - 1\Big)\nabla_\theta\ln\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}).
\end{aligned} \tag{5}$$

However, Equation 4 involves the policy ratio, which can potentially lead to training collapse as discussed in Section 4.2. To mitigate this issue, we clip the policy ratio to be no greater than $c+1$. Importantly, this clipping is not applied directly to the KL divergence estimator in Equation 4, but rather to its gradient (Equation 5). This design ensures that when the ratio exceeds the threshold, the policy drift term continues to provide a gradient that reduces the ratio: when $\frac{\text{sg}(\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}))}{\pi_{\theta_{old}}(y_i|\mathbf{x},\mathbf{y}_{<i})} - 1 > c$,

$$\nabla_\theta\mathcal{E}^i_{\theta_{old},\theta}(\mathbf{x},\mathbf{y}) = c\cdot\nabla_\theta\ln\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i}).$$

In contrast, if clipping were applied to the estimator itself, the resulting gradient $-\nabla_\theta\ln\pi_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})$ would further increase the ratio once it exceeds the threshold, exacerbating training instability.

(III) Weighted advantage: In the view of the response level, each prompt can be viewed as a distinct task. Consequently, we can introduce a per-prompt weighting factor $\omega(\mathbf{x})$ to assign different levels of importance to different prompts. (1) *Equal weight*: when $\omega(\mathbf{x}) = 1$, A_ω^{CPGD} reduces to the original unweighted form. (2) *STD weight*: when $\omega(\mathbf{x}) = 1/\text{std}(\{\mathcal{R}(\mathbf{x},\mathbf{y}^{(k)})\}_k)$, A_ω^{CPGD} is the same as A^{GRPO}. (3) *Clip-filter-like weight*: when $\omega(\mathbf{x}) = \min(c_\omega, \frac{\#\{\mathbf{x}\in\mathcal{D}\}}{\#\{\mathbf{x}\in\mathcal{D}|\text{std}(\{\mathcal{R}_o(\mathbf{x},\mathbf{y}^{(k)})\}_k)\neq 0\}})$, $c_\omega > 0$, similar weighting strategies have also been explored in concurrent work [11], with an analogous effect to online filtering [34], amplifying the gradient contribution of samples with non-zero advantage.

5 Experiments

5.1 Experiments setup

RL baselines, dataset and implementation details. We compare our CPGD with several widely used RL algorithms, including GRPO [1], REINFORCE++ [6] and RLOO [8] on the MMK12 training dataset [15], which contains 15,616 multimodal math problems with verified answers. All RL algorithms use QwenVL2.5-7B as the base model, trained under the same hyperparameters: rollout and training batch sizes of 128, 8 sampled responses per prompt (temperature 1.0), a learning rate of $1e{-}6$, one PPO epoch, and five training episodes. No reference policy constraint is applied during training, and final performance is reported using the last checkpoint. In our system prompt, reasoning steps and final answers are explicitly marked using `<think>` and `<answer>` tags, respectively (see Appendix B).

Benchmarks, model baselines and Overall metric. We evaluate all algorithms on six widely used benchmarks: MathVista (testmini) [16], MathVerse (testmini) [35], MathVision (test) [17], OlympiadBench (EN-OE split) [36], WeMath [37] and MMK12 [15]. MathVista covers visual QA, logic, algebra, and geometry; MathVerse focuses on mathematically grounded visual understanding; and MathVision extends to abstract visual reasoning. OlympiadBench targets graduate-level competition problems, while WeMath enables fine-grained diagnostic analysis via hierarchically annotated tasks. MMK12 provides 500 multiple-choice questions per subject across math, physics, chemistry, and biology for cross-domain performance evaluation.

We also include several multimodal models as baselines. We evaluate open-source models of comparable model size, trained with various strategies, including QwenVL2.5-7B [31], InternVL2.5-8B [38], InternVL2.5-MPO-8B [39], R1-OneVision [40], OpenVLThinker [41], and MM-Eureka [15], which collectively represent the average performance of this model size across the evaluated benchmarks. We further evaluate the leading closed-source models such as GPT-4o [42] and OpenAI-o1 [18] to represent the most outstanding performance that the current state-of-the-art model can achieve on these benchmarks.

To capture overall model performance across N benchmarks, we define an *Overall* metric by normalizing each score against a strong baseline, QwenVL2.5-7B: Overall $:= \frac{1}{N}\sum_{j=1}^{N} X_j / X_j^{\text{Qwen}}$, where X_j and X_j^{Qwen} are the model and baseline scores on benchmark j.

5.2 Main results

Table 1: Performance comparison of various 7B/8B models and leading closed-source models. Top performer is in **bold** and second-best is underlined (excl. OpenAI-o1/GPT-4o).

Model	MathVista	MathVerse	MathVision	Olypamid	WeMath	MMK12	Overall
Leading models							
GPT-4o	63.8	50.2	30.4	35.0	68.8	49.9	1.16
OpenAI-o1	73.9	57.0	60.3	68.0	98.7	73.9	1.83
Similar-size models							
InternVL2.5-8B	64.4	39.5	19.7	12.3	53.5	45.6	0.81
QwenVL2.5-7B	68.2	47.9	25.4	20.2	62.1	53.6	1.00
InternVL2.5-MPO-8B	68.9	35.5	21.5	7.8	53.5	34.5	0.75
R1-Onevision (7B)	64.1	47.1	23.5	17.3	61.8	39.8	0.91
OpenVLThinker (7B)	70.2	47.9	25.3	20.1	64.3	60.6	1.03
MM-Eureka (7B)	73.0	50.3	<u>26.9</u>	20.1	66.1	64.5	1.07
Different RL algorithms on QwenVL2.5-7B							
RLOO	68.6	48.3	23.0	19.5	65.8	61.3	1.01
REINFORCE++	63.9	45.5	18.2	17.8	66.7	64.3	0.96
GRPO	70.3	**51.4**	25.9	18.5	67.4	65.1	1.06
CPGD (clip-filter-like)	<u>73.4</u>	**51.4**	25.9	**21.5**	**70.2**	**67.3**	<u>1.10</u>
CPGD (STD weight)	**74.0**	<u>50.6</u>	**28.3**	<u>21.4</u>	<u>68.3</u>	<u>65.3</u>	**1.11**

Table 1 presents a comprehensive comparison across multiple multimodal mathematical benchmarks. Closed-source models GPT-4o and OpenAI-o1 demonstrate strong performance across all tasks, with o1 achieving the highest scores overall, notably excelling on MathVision (60.3), Olypamid (68.0) and WeMath (98.7), establishing the current performance upper bound. Among similar-size open models, MM-Eureka shows competitive results. MM-Eureka achieves strong results on MathVista (73.0), MathVision (26.9) and a strong result on MMK12 (64.5). However, our proposed CPGD consistently outperforms all similar-size baselines, achieving top or near-leading scores across all benchmarks, reflecting the effectiveness of our proposed RL algorithm.

We further analyze different RL algorithms under the same setting as ours, including the base model, the training dataset, and the hyperparameters. Among baseline methods, GRPO outperforms RLOO and REINFORCE++ on most benchmarks, particularly on MathVerse (51.4) and MathVision (25.9). However, our proposed CPGD method significantly outperforms all baselines, achieving the best performance. Both variants of CPGD (using either clip-filter-like weights or STD-based weights) yield over a +10% improvement in overall performance compared to the base model QwenVL2.5-7B. Notably, CPGD (STD weight) achieves a +21.8% gain on the in-domain benchmark MMK12, and further demonstrates strong generalization with +8.5% and +11.4% improvements on the out-of-distribution benchmarks MathVista and MathVision, respectively. These results demonstrate that CPGD serves as a strong and robust alternative for RL in LM training.

5.3 Ablation study

Table 2: Results of ablation studies. Top performer is in **bold** and second-best is underlined.

Model	MathVista	MathVerse	MathVision	Olypamid	WeMath	MMK12	Overall
CPGD (STD weight)	**74.0**	50.6	**28.3**	21.4	68.3	65.3	**1.11**
Ablation study on the components (using STD weight)							
PG	67.8	42.0	22.5	8.0	58.6	65.9	0.89
PGD	64.2	41.1	20.8	7.5	58.3	**67.3**	0.86
CPG	72.7	**52.3**	27.6	20.8	**70.7**	66.2	**1.11**
Ablation study on the weighting factor							
unprocessed rewards	69.1	40.2	21.8	3.5	59.7	67.2	0.85
equal weight	73.1	51.1	27.2	20.8	67.9	65.8	1.09
clip-filter-like weight	73.4	51.4	25.9	**21.5**	70.2	**67.3**	1.10
Ablation study on the reference constraint (using STD weight)							
w/ reference constraint	71.8	50.0	21.0	21.2	69.8	65.8	1.05

Component ablation. We conduct ablation on key components of our method by comparing variants: PG (basic policy gradient), PGD (PG + policy drift), CPG (PG + clip mechanism), and CPGD. Results show that the clip mechanism plays the most critical role, as seen by the performance drop from CPG/CPGD to PG/PGD across nearly all benchmarks. This aligns with our observation in Section 4.2 that clipping mitigates the response length collapse issue, which otherwise can impair test-time computation and reasoning capabilities. In contrast, adding policy drift has a relatively smaller effect. This is because CPGD's objective lacks a potentially unstable importance-sampling ratio and already benefits from proximal updates via clipping, making policy drift mainly serve as a safeguard against excessive ratio deviation.

Weighting factor ablation. We further ablate different weighting strategies. We additionally include a baseline that uses raw *unprocessed rewards* as advantages, which results in significant performance degradation. This confirms that subtracting the group mean is crucial for stable and effective learning. This approach prevents over-penalization of all responses in the failure cases, which may otherwise trigger a *squeezing effect* [43], where the $\mathrm{Softmax}$ output head unintentionally reallocates probability mass to unexpected tokens, resulting in undesirable behaviors. Both clip-filter-like weight and STD weight outperform equal weighting, which we attribute to their ability to assign greater emphasis to samples with non-zero advantages. This targeted weighting encourages the model to focus more on informative training signals, thereby contributing to the improved performance.

Reference constraint ablation. Removing the reference constraint consistently improves performance, which echoes findings from recent studies [9, 10, 29], suggesting that such constraints may overly restrict policy improvement, and thus hinder overall optimization.

6 Discussion

6.1 Importance sampling

Importance sampling is a valuable technique for correcting the sampling distribution when the learned policy and the behavior policy differ significantly, thereby improving sample efficiency. While we omit the importance-sampling ratio to reduce variance, we **do not** suggest discarding it entirely. In fact, we use a single PPO epoch during training, a widely recommended default [6, 15]. In our view, importance sampling can be omitted with one epoch but should be reintroduced when using more:

$$A_\omega^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) \leftarrow \text{clip}_{1-\epsilon}^{1+\epsilon}\left(\frac{\text{sg}(\pi_{\theta^{(m-1)}}(y_i|\mathbf{x}, \mathbf{y}_{<i}))}{\pi_{\theta_{old}}(y_i|\mathbf{x}, \mathbf{y}_{<i}))}\right) A_\omega^{\text{CPGD}}(\mathbf{x}, \mathbf{y}), \quad m = 1, \ldots, M,$$

where $\pi_{\theta^{(m)}}$ denotes the updated policy after the m-th PPO epoch, and $\pi_{\theta^{(0)}} = \pi_{\theta_{old}}$, and thus the final updated policy is $\theta_{new} = \theta^{(M)}$ after total M epochs. Here, the truncated importance sampling weight is applied to correct the off-policy distribution. Notably, we use $\theta^{(m)}$ rather than the real-time θ to avoid instability caused by frequent updates within a single PPO epoch. This also ensures consistency with our proposed method. However, maintaining $\pi_{\theta^{(m)}}$ may incur additional cost, which we leave for future work to optimize.

6.2 Forward KL divergence vs. reverse KL divergence

Our policy drift adopts the *forward KL divergence* $D_{\text{KL}}(\pi_{\theta_{old}}, \pi_\theta|\mathbf{x})$ instead of the *reverse KL divergence* $D_{\text{KL}}(\pi_\theta, \pi_{\theta_{old}}|\mathbf{x})$. While forward KL has been explored before [12], it is considered less effective than PPO-clip. In contrast, reverse KL is more commonly used in theory because it is closely related to mirror descent and has strong convergence guarantees [44, 45].

Although these two KL forms are different in how they are calculated, they often lead to similar results in practice [46]. This is because both are used to control policy updates. In fact, the difference between their gradients turns out to be small when the policy ratio is small, which is usually the case during training as shown in Figure 1:

$$\nabla_\theta D_{\text{KL}}(\pi_\theta, \pi_{\theta_{old}}|\mathbf{x}) - \nabla_\theta D_{\text{KL}}(\pi_{\theta_{old}}, \pi_\theta|\mathbf{x}) \approx \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{old}}(\cdot|\mathbf{x})}\left[\frac{1}{2}\left(\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right)^2 \nabla_\theta \ln \pi_\theta(\mathbf{y}|\mathbf{x})\right].$$

This approximation holds because $x \ln x \approx x - 1 + \frac{1}{2}(x-1)^2$ when x is close to 1. Despite their similarity, we prefer forward KL for two main reasons: (1) It avoids importance sampling, which reverse KL requires; and (2) It can be cleanly split into per-token terms (see Equation 4), which is not possible with reverse KL due to the importance weights.

6.3 Exploitation vs. exploration

Recent work [47] claims that the performance ceiling of a model is determined by its base model, casting a pessimistic view on the role of RL. While we do not fully agree or disagree, we offer a more nuanced view: *the exploration capability is largely determined by the base model*.

In RL training for LMs, the set of possible responses is constrained by what the base model can generate. RL helps it pick the best ones, boosting metrics like Maj@K. In other words, pretraining and SFT shape what the model can explore, while RL enhances the model's exploitation ability.

This work mainly aims to improve RL stability, but advancing LM reasoning capability requires improving both RL and earlier stages like SFT to expand the model's exploration range. Encouraging active exploration may be key to unlocking further improvements in model performance.

7 Conclusion

We identify a critical source of instability in existing RL methods for LMs: the use of asymmetric clipping on importance-sampling ratios, which can result in training collapse. To address this, we

propose *CPGD*, a principled alternative that avoids direct dependence on policy ratios while enforcing proximal updates through the clip mechanism and policy drift. CPGD further incorporates a stable KL estimator and a weighted advantage strategy to improve learning robustness. Theoretically grounded and empirically validated, CPGD demonstrates superior stability and performance across multimodal math benchmarks, offering a strong and stable RL solution for training LMs.

References

[1] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[2] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022.

[3] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020.

[4] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

[5] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[6] Jian Hu, Jason Klein Liu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025.

[7] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *Deep Reinforcement Learning Meets Structured Prediction, ICLR 2019 Workshop*. OpenReview.net, 2019.

[8] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 12248–12267. Association for Computational Linguistics, 2024.

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

[10] 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, 2025.

[11] Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. Gpg: A simple and strong reinforcement learning baseline for model reasoning, 2025.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

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

[14] John Schulman. Approximating kl divergence, 2020. *URL http://joschu. net/blog/kl-approx. html*, 2023.

[15] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning, 2025.

[16] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts, 2024.

[17] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset, 2024.

[18] OpenAI. Introducing openai o1. https://openai.com/o1/, 2024. Accessed: 2024-10-02.

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

[20] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, 2025.

[21] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025.

[22] Qwen Team. Qvq: To see the world with wisdom, December 2024.

[23] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025.

[24] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[25] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[26] YingZhe Peng, Gongrui Zhang, Xin Geng, and Xu Yang. Lmm-r1. https://github.com/TideDra/lmm-r1, 2025. Accessed: 2025-02-13.

[27] Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. https://github.com/Deep-Agent/R1-V, 2025. Accessed: 2025-02-02.

[28] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning, 2025.

[29] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, and Heung-Yeung Shum Xiangyu Zhang. Open-reasoner-zero: An open source approach to scaling reinforcement learning on the base model. https://github.com/Open-Reasoner-Zero/Open-Reasoner-Zero, 2025.

[30] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6672–6679, 2020.

[31] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[32] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework, 2024.

[33] 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, 2024.

[34] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025.

[35] Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems?, 2024.

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

[37] Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong, Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. We-math: Does your large multimodal model achieve human-like mathematical reasoning?, 2024.

[38] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, 2025.

[39] Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization, 2024.

[40] Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, Bo Zhang, and Wei Chen. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization, 2025.

[41] Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. Openvlthinker: An early exploration to complex vision-language reasoning via iterative self-improvement, 2025.

[42] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card, 2024.

[43] Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *International conference on machine learning*, pages 2160–2169. PMLR, 2019.

[45] Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5668–5675, 2020.

[46] Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. Revisiting design choices in proximal policy optimization, 2020.

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

Appendix

A Proofs

A.1 Proof for Proposition 1

Proposition 2. *Let θ_0 be a parameter such that the importance-sampling ratio satisfies $|\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1| = \epsilon$. Consider updating θ_0 using either (i) the PPO-clip objective, resulting in parameter θ_1^{PPO}, or (ii) the CPGD objective with $\alpha = 0$, yielding parameter θ_1^{CPG}. Then, there exists a constant $\eta_{\max} > 0$ such that for any learning rate $\eta \in (0, \eta_{\max})$, the following inequality holds:*

$$\left|\frac{\pi_{\theta_1^{PPO}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| > \left|\frac{\pi_{\theta_1^{CPG}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| > \left|\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1\right| = \epsilon.$$

After one update step, both PPO and CPG increase the importance-sampling ratio deviation from the old policy, but PPO does so more aggressively than CPG.

Proof. Consider $f(\eta) = \frac{\pi_{\theta_1^{\text{CPG}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}$, where $\theta_1^{\text{CPG}} = \theta_0 + \eta \nabla_\theta \hat{\mathcal{L}}_{\text{CPG}}(\mathbf{x}, \mathbf{y}; \theta_0)$ is the single gradient ascent step on the empirical CPGD objective (Equation 1) without the policy drift term. The gradient of the objective takes the form:

$$\nabla_\theta \hat{\mathcal{L}}_{\text{CPG}}(\mathbf{x}, \mathbf{y}; \theta) = A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) \nabla_\theta \ln \pi_\theta(\mathbf{y}|\mathbf{x}).$$

Thus, for the case where $\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} = 1 + \epsilon$ and $A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) > 0$, the directional derivative of f at $\eta = 0$ satisfies:

$$f'(0) = \langle \frac{\nabla_\theta \pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}, \nabla_\theta \hat{\mathcal{L}}_{\text{CPG}}(\mathbf{x}; \theta_0) \rangle > 0.$$

Hence, there exists a constant $\eta_1 > 0$ such that for any $\eta \in (0, \eta_1)$, we have $f(\eta) > f(0)$. Similarly, when $\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} = 1 - \epsilon$ and $A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) < 0$, there exists $\eta_2 > 0$ such that $f(\eta) < f(0)$ for any $\eta \in (0, \eta_2)$.

Therefore, for any $0 < \eta < \min(\eta_1, \eta_2)$, the following holds:

$$|\frac{\pi_{\theta_1^{\text{CPG}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1| > |\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1| = \epsilon. \tag{6}$$

Next, define $g(\eta) = \frac{\pi_{\theta_1^{\text{CPG}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - \frac{\pi_{\theta_1^{\text{PPO}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}$, where $\theta_1^{\text{PPO}} = \theta_0 + \eta \nabla_\theta \hat{\mathcal{L}}_{\text{PPO}}(\mathbf{x}, \mathbf{y}; \theta_0)$ and

$$\nabla_\theta \hat{\mathcal{L}}_{\text{PPO}}(\mathbf{x}, \mathbf{y}; \theta) = A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) \frac{\nabla_\theta \pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}.$$

For the case where $\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} = 1 + \epsilon$ and $A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) > 0$, we have:

$$g'(0) = \left\langle \frac{\nabla_\theta \pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}, A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) \cdot (1 - \frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})}) \cdot \nabla_\theta \ln \pi_\theta(\mathbf{y}|\mathbf{x}) \right\rangle < 0.$$

Hence, there exists a constant $\eta_3 > 0$ such that $g(\eta) < g(0)$ for any $\eta \in (0, \eta_3)$. Similarly, for the case where $\frac{\pi_{\theta_0}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} = 1 - \epsilon$ and $A^{\text{CPGD}}(\mathbf{x}, \mathbf{y}) < 0$, there exists a constant $\eta_4 > 0$ such that $g(\eta) > g(0)$ for any $\eta \in (0, \eta_4)$.

Therefore, for any $0 < \eta < \min(\eta_3, \eta_4)$, we have

$$|\frac{\pi_{\theta_1^{\text{PPO}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1| > |\frac{\pi_{\theta_1^{\text{CPG}}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}|\mathbf{x})} - 1|. \tag{7}$$

Therefore, by letting $\eta_{\max} = \min(\eta_1, \eta_2, \eta_3, \eta_4)$, the proof is complete. □

A.2 Proof for Theorem 1

Theorem 2. *Let $\{\pi_{\theta_k}\}_{k=0}^{\infty}$ denote the sequence of policies generated by the CPGD update rule (Equation 1). Then, the sequence π_{θ_k} converges.*

Proof. First, denote $\mathcal{L}_{\text{CPGD}}(\theta;\theta_k) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[g(\theta;\theta_k,\mathbf{x})\big]$, and rewrite g as

$$
\begin{aligned}
g(\theta;\theta_k,\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\mathcal{R}_o(\mathbf{x},\mathbf{y})\ln\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})}\Big] - \alpha D_{\text{KL}}(\pi_{\theta_k},\pi_\theta|\mathbf{x}) \\
-\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\text{ReLU}\Big(\big[\ln\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})} - \text{clip}\big(\ln\frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})},\ln(1-\epsilon),\ln(1+\epsilon)\big)\big]\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big)\Big].
\end{aligned}
$$

Here, we omit the baseline $\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}[\mathcal{R}_o(\mathbf{x},\mathbf{y})]$. Then, denoting θ_{k+1} the point such that $\mathcal{L}_{\text{CPGD}}(\theta_{k+1};\theta_k) \geq \mathcal{L}_{\text{CPGD}}(\theta_k;\theta_k)$, we obtain

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_{k+1}}(\cdot|\mathbf{x})}\Big[\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big] - \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big] \\
=&\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\big(\frac{\pi_{\theta_{k+1}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})} - 1\big)\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big] \\
\geq&\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\ln\frac{\pi_{\theta_{k+1}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})}\cdot\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big] \\
=&g(\theta_{k+1};\theta_k,\mathbf{x}) - g(\theta_k;\theta_k,\mathbf{x}) + \alpha D_{\text{KL}}(\pi_{\theta_k},\pi_{\theta_{k+1}}|\mathbf{x}) \\
&+\mathbb{E}_{\mathbf{y}\sim\pi_{\theta_k}(\cdot|\mathbf{x})}\Big[\text{ReLU}\Big(\big[\ln\frac{\pi_{\theta_{k+1}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})} - \text{clip}\big(\ln\frac{\pi_{\theta_{k+1}}(\mathbf{y}|\mathbf{x})}{\pi_{\theta_k}(\mathbf{y}|\mathbf{x})},\ln(1-\epsilon),\ln(1+\epsilon)\big)\big]\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big)\Big].
\end{aligned}
$$

Denoting the overall expected return by $\eta(\pi_\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\Big[\mathcal{R}_o(\mathbf{x},\mathbf{y})\Big]$, we integrate over $\mathbf{x}$ to conclude

$$
\eta(\pi_{\theta_{k+1}}) - \eta(\pi_{\theta_k}) \geq \alpha\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[D_{\text{KL}}(\pi_{\theta_k},\pi_{\theta_{k+1}}|\mathbf{x})\Big] \overset{\text{Pinsker inequality}}{\geq} \frac{\alpha}{2}\|\pi_{\theta_{k+1}} - \pi_{\theta_k}\|_1^2.
$$

Because $\eta(\pi_{\theta_k})$ is bounded, there exists a η_* such that $\lim_{k\to\infty}\eta(\pi_{\theta_k}) = \eta_*$. Thus, taking the limit of k on both sides of the following equation,

$$
0 \leq \frac{\alpha}{2}\|\pi_{\theta_{k+1}} - \pi_{\theta_k}\|_1^2 \leq \eta(\pi_{\theta_{k+1}}) - \eta(\pi_{\theta_k}),
$$

we can obtain $\lim_{k\to\infty}\|\pi_{\theta_{k+1}} - \pi_{\theta_k}\|_1 = 0$. Since the parameter space Θ is compact, the sequence $\{\pi_{\theta_k}\}$ converges to some limit point π_{θ_*}.

□

B Prompt setting

Table 3: Prompt setting.

SYSTEM: Solve the question. The user asks a question, and you solves it. You first thinks about the reasoning process in the mind and then provides the user with the answer. The answer is in latex format and wrapped in $...$. The final answer must be wrapped using the \boxed{} command. The reasoning process and answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>Since $1+1=2$, so the answer is 2. </think><answer>The answer is $\boxed{2}$ </answer>, which means the final answer assistant's output should start with <answer> and end with </answer>. **USER:** <image>{{question}}

We follow the prompt format from DeepSeek-R1, where reasoning steps and final answers are explicitly marked using `<think>` and `<answer>` tags, respectively. The full prompt template is provided in Table 3.

C Limitations

While this work introduces a stable and effective RL method for LMs training, it has several limitations: (1) For the weighted advantage component, we conducted only preliminary experiments and did not thoroughly explore the impact of different weighting factors. Our results suggest that using non-uniform weights yields better performance than trivial equal weighting, but further investigation is needed. (2) Our study focuses on on-policy training; we leave off-policy settings—where importance sampling is typically required—for future work. Ensuring training stability in the presence of importance sampling remains an open question. (3) All experiments were conducted on standard academic-scale models (7B parameters). We did not evaluate our method on larger models (e.g., 100B+), which would require significant computational resources.