EFFICIENT POLICY-SPACE REGRESSION VIA REPARAMETERIZED OPTIMAL TARGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

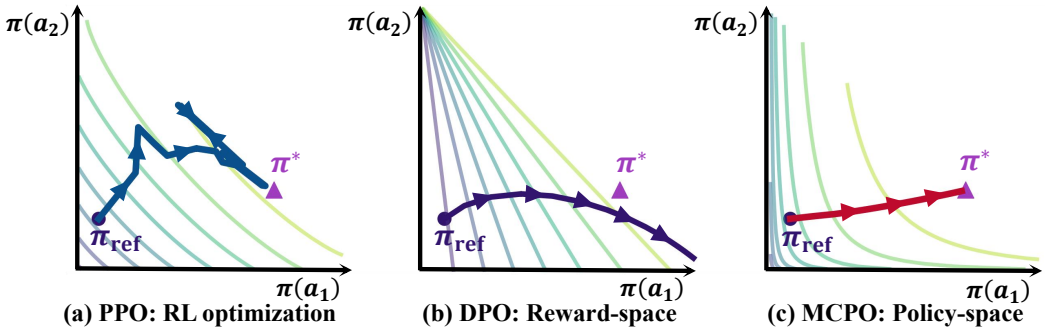
Supervised ranking-based methods such as Direct Preference Optimization (DPO) have shown that bypassing complex RL dynamics can improve the stability and efficiency of post-training. However, these methods impose supervision primarily in the reward space, and their guarantees do not generally ensure alignment of the model policy with the optimal policy in policy space, potentially introducing bias. To address this issue, we propose **M**inimum **C**ross-entropy **P**olicy **O**ptimization (MCPO), which performs regression directly in the policy space. MCPO first **reparameterizes the optimal policy** into a more tractable target, and then **regresses** the policy toward it using a low-variance cross-entropy-style objective. Theoretically, we show that the optimal policy remains a stationary point of the MCPO objective, preserving the desired optimum under practical variance control. Moreover, under idealized tabular softmax models, MCPO guarantees monotonic progress toward the optimal policy when clipping is non-degenerate; without clipping, it further yields the closest single finite-step update to the optimal policy among all objectives. Empirically, MCPO consistently outperforms all baselines across all evaluated settings. In preference alignment, it achieves better performance using only 50% of the training steps, establishing an efficient paradigm for LLM post-training.

1 INTRODUCTION

Reinforcement learning (RL) has become the dominant paradigm for post-training large language models (LLMs), enabling alignment with human preferences (Achiam et al., 2023; Dubey et al., 2024; Comanici et al., 2025; Bai et al., 2023) and the acquisition of advanced reasoning capabilities (Guo et al., 2025; Team et al., 2025). Despite its empirical success, standard PPO-style RL optimization (Schulman et al., 2017; Ouyang et al., 2022; Shao et al., 2024) remains inherently inefficient and unstable. As illustrated in Figure 1(a), conventional RL methods such as PPO (Achiam et al., 2023) update the policy **along the policy-gradient direction**, which produces the fastest local improvement, but can deviate substantially from the direction towards the globally optimal policy. This mismatch is particularly pronounced in the early stage of training and when the reward gaps among some candidate actions are small, causing the policy to follow a highly tortuous update trajectory and requiring a substantial number of optimization steps. Moreover, mainstream RL objectives typically clip the reward-related term while leaving the KL regularization term unclipped. In the late stage of training, once the reward term is clipped, the comparatively strong KL penalty can disrupt the balance of optimization and thereby induce severe instability. These limitations naturally raise the following question: can we obtain a more stable and direct policy optimization approach?

A common strategy is to transform the RL objective into a stable and direct supervised learning objective (Zhao et al., 2023). However, this introduces a fundamental difficulty: although the theoretical optimal policy admits a closed-form expression (Rafailov et al., 2023), computing it exactly requires enumerating all actions in the partition function. Such computation is infeasible in the LLM setting, where the action space spans the entire natural language space. To address this challenge, methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) substitute the analytical form of the optimal policy into a loss of the reward-model, thus eliminating the intractable partition function. This idea underlies nearly all existing approaches that bypass complex RL dynamics (Meng et al., 2024; Gao et al., 2024; Hong et al., 2024; Brantley et al., 2025). Nevertheless, while these methods put training into a supervised form, **the supervision is imposed in the reward space** rather than the policy space. Consequently, they can typically guarantee convergence only for

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.



(a) PPO: RL optimization dynamics **(b) DPO: Reward-space supervision** **(c) MCPO: Policy-space supervision**

Figure 1: Simulated convergence trajectories in a 3-action space, projected onto the $\pi(a_1)$-$\pi(a_2)$ plane. In this case, PPO iterates toward the optimal policy π^* along a tortuous and unstable trajectory, DPO converges stably but toward a greedy policy, while MCPO converges to π^* along an almost straight trajectory.

implicit reward (Rafailov et al., 2023; Gao et al., 2024), whereas convergence of the policy itself generally does not follow (Liu et al., 2024; Han et al., 2025). A direct manifestation of this mismatch is severe overfitting (Azar et al., 2024): as shown in Figure 1(b), the policy may move rapidly toward the optimum at the beginning of training, yet ultimately fails to converge to the true optimal policy. These observations suggest that supervision outside the policy space is fundamentally insufficient, and that a more principled solution should perform supervision directly in the policy space.

To resolve this, we seek an alternative way to address the intractability of the optimal policy. We theoretically reveals a key property: although the optimal policy cannot be computed exactly, the partition-function estimation error is identical across actions under the same state. Under a partially supervised loss, replacing the partition function with any positive action-independent constant is, therefore, equivalent to weighted regression and leaves the per-state convergence behavior unchanged. Motivated by this, we propose **M**inimum **C**ross-Entropy **P**olicy **O**ptimization (MCPO), a novel framework that achieves efficient reinforcement learning by directly regressing toward a reparameterized target. **This target replaces the partition function with an action-independent constant and can therefore be viewed as a reparameterization of the optimal policy.** This makes it exactly computable while preserving both the unclipped policy-gradient direction and the stationary policy at each state. MCPO further combines a cross-entropy-style objective with clipping (Schulman et al., 2017) and baseline (Williams, 1992) techniques for stable policy regression.

Our theoretical analysis **first** shows that MCPO preserves the optimal policy as a stationary policy under gradient-based optimization, thereby mitigating the policy-space bias associated with reward-space regression methods. **Following prior works** (Cui et al., 2025; Chen et al., 2026), we **further** study its convergence properties using **tabular** softmax policy updates as an **idealized model of LLM RL**. Our analysis shows that even when clipping is active, every MCPO update moves the tabular policy closer to the optimum, whereas standard PPO-style RL update does not always guarantee such monotonic progress. Moreover, when clipping is inactive, a single finite-step update under tabular MCPO moves the policy closest to the optimal policy among all objectives. These properties potentially endow MCPO with high optimization efficiency and a more direct, nearly straight-line convergence behavior, as illustrated by the simple example in Figure 1(c). Empirically, we evaluate MCPO on preference alignment, reasoning training, and offline RL. MCPO matches strong baselines such as GRPO using only $\sim$50% of the training steps and consistently outperforms all baselines when trained for the same number of steps.

Our contributions are summarized as follows:

- We propose MCPO, which reformulates RL fine-tuning as a regression task in the policy space, effectively bridging the stability of supervised learning with the optimality of RL.
- We analyze the convergence of MCPO under the same idealized framework adopted in prior work (Cui et al., 2025; Chen et al., 2026), theoretically establishing its advantages in unbiasedness and efficiency.

- MCPO consistently outperforms strong baselines across RLHF, RLVR, and offline RL scenarios, substantially improving the training efficiency of RL for LLMs.

2 Related Work

The goal of reinforcement learning for LLMs is to learn an optimal response-generation policy. Proximal Policy Optimization (PPO) (Schulman et al., 2017), originally developed for traditional deep RL, was among the earliest RL algorithms adopted for LLM alignment (Achiam et al., 2023). At each iteration, policy updates are jointly driven by action rewards and local KL divergence, while a clipping mechanism is typically used to discard samples that deviate substantially from the sampling policy. Subsequent variants, including GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), DAPO (Yu et al., 2026), and Dr-GRPO (Liu et al., 2025), largely preserve these optimization dynamics and therefore inherit their associated limitations, including slow convergence and instability (Zheng et al., 2025). In contrast to these advantage-guided gradient-based methods, MCPO explicitly optimizes the policy through direct regression in the policy space, thereby substantially improving both stability and convergence speed.

Direct Preference Optimization (DPO) (Rafailov et al., 2023) and its variants, such as SimPO (Meng et al., 2024), BCO (Jung et al., 2025), REBEL (Gao et al., 2024), and A*PO (Brantley et al., 2025), represent a paradigm shift toward stability by reframing RL as a supervised learning problem. These methods derive implicit rewards from the model policy and then align the implicit rewards with the true rewards using contrastive losses or direct regression, thereby transforming RL dynamics into regression in the reward space. Such regression improves training stability and reduces the implementation complexity of RL. However, reward-space regression makes alignment in the policy space difficult to guarantee; under conditions such as a suboptimal behavior policy (Liu et al., 2024) or finite samples (Han et al., 2025), the optimal policy may not even be a stationary point. As a result, these methods often suffer from bias in the policy space, which can lead to overfitting (Azar et al., 2024). In contrast to these reward-space regression paradigms, MCPO performs supervision directly in the policy space, thereby guaranteeing mathematical unbiasedness.

3 Preliminary

Reinforcement Learning for LLMs. When using RL for post-training of LLMs, it is generally formulated that each prompt is a state s, and each generated response is an action a. The distribution of a taken by LLMs in each state s is defined as π_θ, which is determined by the model parameters θ and is the target of optimization during post-training. Each pair (s, a) is assigned a task-specific reward $r(a, s)$, which is typically derived from a reward model (in RLHF) or is based on correctness verification (in RLVR). The LLM's policy is expected to maximize the expected reward $\mathbb{E}_{\pi_\theta}[r]$ while constraining the deviation from the reference policy π_{ref} to prevent catastrophic forgetting (Rafailov et al., 2023). Therefore, RL's objective function $\mathcal{L}$ that the LLM needs to maximize can be expressed as:

$$\mathcal{L}(\pi) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{\pi_\theta}[r(a,s)] - \beta\,\mathbb{D}_{\text{KL}}\left(\pi_\theta(\cdot|s)||\pi_{\text{ref}}(\cdot|s)\right)\right], \tag{1}$$

where $\mathcal{D}$ is the dataset, π_{ref} is the reference policy, typically derived from a pre-trained or fine-tuned LLM. β is a hyperparameter that controls the strength of the constraint from π_{ref}. $\mathbb{D}_{\text{KL}}$ represents the KL divergence. MCPO adopts this standard optimization objective, treating the training goal as the maximization of Equation 1.

Optimal Policy. The optimal policy π^* that maximizes Equation 1 can be explicitly derived as (Rafailov et al., 2023; Liu et al., 2024; Ethayarajh et al., 2024):

$$\pi^*(a|s) = \frac{1}{Z(s)}\exp\left(\frac{r(a,s)}{\beta}\right)\pi_{\text{ref}}(a|s), \tag{2}$$

where $Z(s)$ is the partition function, given by:

$$Z(s) = \mathbb{E}_{a'\sim\pi_{\text{ref}}}\left[\exp\left(\frac{r(a',s)}{\beta}\right)\right]. \tag{3}$$

Computing the partition function exactly is computationally intractable due to the enormous action space, making it difficult to accurately estimate the optimal policy as a regression target.

4 Motivation: Direct Updates Toward the Optimal Policy

The key to developing an efficient policy optimization algorithm is to identify an objective that drives the policy toward the optimal policy π^* as rapidly as possible. Multiple objectives may ultimately converge to the same optimal policy, while exhibiting substantially different optimization dynamics and convergence rates. For example, RL dynamics greedily follow the direction of maximal local reward improvement at each step, but this direction does not necessarily move the policy most directly toward the global optimum. Following the common convergence analysis that measures progress by the reduction in distance to the convergence endpoint (Bertsekas, 1997), we seek an update direction that maximizes this one-step improvement. Although this idea is intuitive, directly analyzing convergence at the parameter level of LLMs is highly challenging. Therefore, following prior works (Cui et al., 2025; Chen et al., 2026), we adopt the tabular softmax policy as an idealized model and establish the following lemma:

Lemma 1. *Under a tabular softmax policy optimized with natural policy gradient, for a sufficiently small common step size, using the cross-entropy with either the optimal policy π^* or a reparameterized optimal target $\hat{\pi}^* = \rho\pi^*, \rho > 0$ as the objective maximizes the one-step progress toward π^* among all objective functions, thereby inducing the most direct update toward the optimal policy.*

Lemma 1 shows that, at least in the tabular softmax setting, directly regressing toward the optimal policy via cross-entropy in policy space provides the most direct and effective update, as each iteration moves the policy closer to the optimum than updates induced by other objectives. Although this lemma is established only in the tabular setting, LLM policies and tabular policies exhibit similar behavior under several macroscopic convergence metrics (Cui et al., 2025). We therefore use this result as motivation for designing an LLM policy optimization method based on cross-entropy regression toward the optimal policy in policy space.

5 Minimum Cross-Entropy Policy Optimization

5.1 Reparameterized Optimal Target

Definition. Since the optimal policy π^* cannot be computed exactly, MCPO introduces an alternative regression target that is easy to compute while still guiding the policy toward π^*. We identify a practical formulation, termed the reparameterized optimal target $\hat{\pi}^*$. Compared with Equation 2, the only difference between π^* and $\hat{\pi}^*$ lies in the definition of the partition function:

$$\widehat{Z}(s) = \exp\left(\frac{\mathrm{Avg}_{a\in\mathcal{B}_s}\left(r(a,s)\right)}{\beta}\right), \tag{4}$$

where $\widehat{Z}(s)$ denotes the partition function used by the reparameterized optimal target, and $\mathrm{Avg}_{a\in\mathcal{B}_s}$ denotes the average over all samples associated with the same prompt s within a batch $\mathcal{B}$. Accordingly, the reparameterized optimal target $\hat{\pi}^*$ is defined as

$$\hat{\pi}^*(a|s) = \frac{1}{\widehat{Z}(s)} \exp\left(\frac{r(a,s)}{\beta}\right) \pi_{\text{ref}}(a|s), \tag{5}$$

which serves as the target in the cross-entropy objective of MCPO.

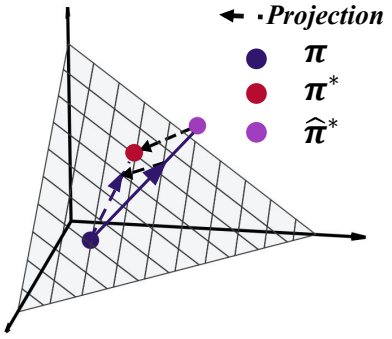


Figure 2: The relationship between the reparameterized optimal target $\hat{\pi}^*$ and the optimal policy π^*. Under the KL divergence, projecting the model policy π_θ toward $\hat{\pi}^*$ onto the policy space corresponds to moving it toward π^*.

Properties. Clearly, $\hat{\pi}^*$ **does not necessarily sum to one** over all actions. Therefore, it is **typically not** a valid policy distribution and cannot be used directly for response generation. Nevertheless, the discrepancy between $\widehat{Z}(s)$ and $Z(s)$ is identical across all actions under the same state. As illustrated in Figure 2, π^* can therefore be viewed as the projection of $\hat{\pi}^*$ onto the policy space, implying that moving π toward $\hat{\pi}^*$ also drives it toward π^*. Using $\hat{\pi}^*$ in the cross-entropy-style objective still guarantees unbiased convergence at the state level, making it a valid convergence target.

5.2 POLICY UPDATE OBJECTIVES

Definition. The most natural objective for regressing toward the reparameterized optimal target is to directly compute the cross-entropy between the model policy π_θ and $\hat{\pi}^*$, which is given by:

$$\mathcal{L}_{\mathrm{CE}}(\pi_\theta, \hat{\pi}^*) = -\mathbb{E}_{\hat{\pi}^*}\left[\log \pi_\theta(a|s)\right] = -\mathbb{E}_{\pi_{\text{old}}}\left[\frac{\hat{\pi}^*(a|s)}{\pi_{\text{old}}(a|s)} \log \pi_\theta(a|s)\right], \tag{6}$$

where we extend the domain of the expectation operator $\mathbb{E}_{p(X)}[f(X)] = \sum_X p(X) f(X)$ to all nonnegative p. Although unbiased, Equation 7 can have high variance when $\hat{\pi}^*$ differs substantially from π_{old}, resulting in large errors when estimated directly via Monte Carlo sampling. Therefore, noting that $\mathbb{E}_{\pi_\theta}[k \nabla \log \pi_\theta] = 0$, we first employ a baseline technique (Hu et al., 2025) to reduce the variance without introducing bias:

$$\mathcal{L}_{\mathrm{CE-b}}(\pi_\theta, \hat{\pi}^*) = -\mathbb{E}_{\pi_{\text{old}}}\left[\frac{\mathrm{sg}[\pi_\theta(a|s)]}{\pi_{\text{old}}(a|s)} \left\{\frac{\hat{\pi}^*(a|s)}{\mathrm{sg}[\pi_\theta(a|s)]} - 1\right\} \log \pi_\theta(a|s)\right], \tag{7}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, and it is clearly that $\nabla \mathcal{L}_{\mathrm{CE-b}}(\pi_\theta, \hat{\pi}^*) = \nabla \mathcal{L}_{\mathrm{CE}}(\pi_\theta, \hat{\pi}^*)$. To further reduce the variance, we additionally apply clipping to avoid excessively large importance weights, yielding the final MCPO objective:

$$\mathcal{L}_{\mathrm{MCPO}}(\pi_\theta) = -\mathbb{E}_{\pi_{\text{old}}}\left[\frac{\mathrm{sg}[\pi_\theta(a|s)]}{\pi_{\text{old}}(a|s)} \mathrm{clip}\left\{\frac{\hat{\pi}^*(a|s)}{\mathrm{sg}[\pi_\theta(a|s)]} - 1, \tau_l, \tau_u\right\} \log \pi_\theta(a|s)\right], \tag{8}$$

where τ_l and τ_u denote the clipping thresholds. Crucially, the clipping mechanism is applied only to the weighting coefficient and is completely detached from the differentiable term $\log \pi_\theta(a|s)$. This structural decoupling prevents the optimization instability that would otherwise arise from inadvertently zeroing out part of the gradients.

Properties. The gradients of $\nabla \mathcal{L}_{\mathrm{MCPO}} = \nabla \mathcal{L}_{\mathrm{CE}}$ are guaranteed to coincide only when clipping is inactive, the MCPO gradient can be biased during the early stages of training. However, we will show later that they share the same stationary points, and hence MCPO is unbiased in terms of optimality, resolving the bias in reward-space regression.

5.3 THEORETICAL ANALYSIS

Unbiased Stationary Point. Although the gradients induced by $\mathcal{L}_{\mathrm{MCPO}}$ and $\mathcal{L}_{\mathrm{CE}}$ are not necessarily identical, both objectives admit the optimal policy π^* as a stationary point, making MCPO unbiased from an optimization perspective, as formalized by the following theorem.

Theorem 1. *Under bounded rewards and $\beta > 0$, the optimal policy π^* is a stationary point of Equation 8.*

Theorem 1 shows that although the target $\hat{\pi}^*$ generally differs from the true optimal policy π^*, the learned policy is still guaranteed to stop updating at π^*.

Monotonic Improvement in the Tabular Setting. Although we have established that the convergence endpoint is unbiased, a more detailed analysis of the convergence behavior of MCPO remains valuable. However, directly analyzing the optimization dynamics at the parameter level of LLMs is highly challenging. We therefore again adopt the tabular softmax policy as an idealized model and establish the following theorem:

Theorem 2. *Assume that the clipped weights are not all saturated at the same clipping boundary, i.e., they are not uniformly truncated to τ_l for all actions, nor uniformly truncated to τ_u for all actions. Let $\eta > 0$ denote the step size, π the current policy, π^* the optimal policy, π_{ref} the reference policy, and r the reward function. Consider the updated tabular softmax policies π_{MCPO} and π_{PPO} obtained after a single iteration using MCPO and the standard PPO-style objective, respectively. Let the one-step improvement toward optimality be defined as $\Delta(\cdot) = \mathbb{D}_{\mathrm{KL}}(\pi^* \| \pi) - \mathbb{D}_{\mathrm{KL}}(\pi^* \| \cdot)$. Then the following properties hold: (1) There always exists a step size $\eta > 0$ such that the MCPO update guarantees a strictly positive improvement, i.e., $\Delta(\pi_{\mathrm{MCPO}}) > 0$; (2) In contrast, under certain feasible conditions, achieving positive improvement $\Delta(\pi_{\mathrm{PPO}}) > 0$ is impossible regardless of the choice of η.*

Theorem 2 characterizes the monotonic convergence behavior of tabular MCPO. Even with clipping, tabular MCPO can still guarantee monotonic progress toward the optimal policy, whereas PPO-style RL with clipping cannot provide the same guarantee. This highlights the advantage of MCPO under the idealized tabular setting.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Preference Alignment Experimental Setup. *(1) Training.* Similar to related works (Achiam et al., 2023; Li et al., 2023b; Ahmadian et al., 2024), we take UltraFeedback (Cui et al., 2023) as dataset and use a reward model (Yang et al., 2024) trained on human preference data to calculate rewards and train the LLM using online reinforcement learning. Base models include Meta-Llama3.1-7B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Qwen2.5-3B-Instruct (Bai et al., 2023), and Qwen2.5-1.5B-Instruct (Bai et al., 2023). *(2) Evaluation.* Following the related work (Meng et al., 2024; Han et al., 2025; Li et al., 2023b; Ahmadian et al., 2024), we evaluated the LLMs on two benchmarks, AlpacaEval (Dubois et al., 2024) and Arena-Hard (Li et al., 2024), which show strong alignment with human preferences. Specifically, we used the officially (Li et al., 2023a) supported Meta-Llama-3-70B-Instruct (Dubey et al., 2024) model as the judge model for AlpacaEval and the GRM-Llama3.1-8B-rewardmodel-ft (Yang et al., 2024) as the judge model for Arena-Hard, which is reported to achieve over 90% agreement with human preferences. *(3) Baselines.* Our baseline selection covers the full spectrum of modern online RLHF approaches to ensure a rigorous comparison. We include PPO (Ouyang et al., 2022), GRPO (Shao et al., 2024), BNPO (Xiao et al., 2025), Dr-GRPO (Liu et al., 2025), DPH-F, and DPH-JS (Li et al., 2026) as standard reinforcement learning baselines, and use standard online DPO (Rafailov et al., 2023), IPO (Azar et al., 2024), and REBEL (Gao et al., 2024) as supervised regression algorithms in the reward space. All baselines were implemented using the official TRL library (von Werra et al., 2020) to guarantee reproducibility. We include the details of parameters such as the learning rate, β and group size in Appendix D.1.

RLVR Experimental Setup. We utilized KW-R1 (Liu & Zhang, 2025) to implement RLVR, employing the correctness of reasoning results as the reward to train the Qwen2.5-7B (Bai et al., 2023) model. *(1) Math.* For the mathematical task, we trained the model using the GSM8k (Cobbe et al., 2021) training set, guiding it to autonomously generate a chain of thought labeled with $< think >$, and set rewards based on the accuracy of the final answer. The model's performance was ultimately evaluated on the GSM8k test set using the pass@1 metric. *(2) Coding.* For the coding task, the model was trained on the Code-R1 (Liu & Zhang, 2025) training set. Rewards were determined based on several criteria: whether the code generated after reasoning could be executed, passed all test cases, produced correct results, and whether its memory usage and runtime fell within specified limits. The model was evaluated on the Code-R1 test set using pass@1, pass@3, and pass@5 metrics, which measure the proportion of problems solved by the model in several reasoning attempts. *(3) Baselines.* To ensure a rigorous evaluation, we benchmark MCPO against a comprehensive suite of methods, including Reinforce++ (Hu et al., 2025) and GRPO as the current popular for reasoning tasks. Additionally, we include recent advanced variants such as Dr-GRPO (Liu et al., 2025) and DPH (Li et al., 2026), which are designed to further enhance alignment stability and performance. In addition, we compare MCPO with A*PO (Brantley et al., 2025), a reward-space regression method in the RLVR setting. More details about training and evaluation is in Appendix D.2.

Offline RL Experimental Setup. *(1) Training.* Aligning with previous work (Ethayarajh et al., 2024), we employed the UltraFeedback (Cui et al., 2023) dataset for offline training. In our setup, we utilized the original high-quality human annotations from UltraFeedback directly. To ensure a fair comparison with baselines that lack access to scalar reward values (relying solely on preference pairs), MCPO assigns a +1 reward to positive samples and a -1 reward to negative ones, avoiding any privileged information. *(2) Evaluation.* To provide a comprehensive evaluation of general capabilities, we tested on a diverse suite of benchmarks: GSM8k for mathematical proficiency (Math); ARC-Easy, ARC-Challenge, and PIQA for logical reasoning (Reasoning); and MMLU and GPQA for broad knowledge acquisition (Knowledge). Crucially, this multi-dimensional assessment allows us to investigate potential **biases** introduced by alignment algorithms—specifically, whether optimizing for preferences in one domain inadvertently degrades performance in others. All evaluations are conducted using llama-factory (Zheng et al., 2024) and evalscope (Team, 2024). *(3) Baselines.* We compare MCPO against a wide spectrum of offline baselines, including DPO as the standard reference, margin-based methods like SLiC (Zhao et al., 2023) and IPO, and recent state-of-the-art approaches such as RPO (Pang et al., 2024), BCO (Jung et al., 2025), and simPO (Meng et al., 2024), ensuring a rigorous comparison across different loss formulations. All baseline methods was implemented via TRL. Further details are provided in Appendix D.3.

Table 1: Experimental results on preference alignment. Here, 'Base" refers to the base model, "R-Reg." denotes reward-space regression, "RL-Dyn." denotes optimization using RL dynamics, and "P.-Reg." denotes policy-space regression. **Bold** indicates the best performance. On AlpacaEval, WR_1 and WR_2 denote the Win Rates on versions 1.0 and 2.0, respectively, while LC_2 denotes the Length-Controlled Win Rate on v2.0. For Arena Hard, we report Win Rates against GPT-4-0613 (v0.1) and Gemini-2.0-Flash (v2.0), denoted as GPT and Gemini, respectively.

Type	Method	Llama3.1-8B-Instruct					Mistral-7B-Instruct				
		Alpaca Eval			Arena Hard		Alpaca Eval			Arena Hard	
		WR_2 ↑	LC_2 ↑	WR_1 ↑	GPT ↑	Gemini ↑	WR_2 ↑	LC_2 ↑	WR_1 ↑	GPT ↑	Gemini ↑
-	Base	42.13	46.72	92.29	36.20	17.92	18.52	27.61	93.11	25.40	13.50
RL-Dyn.	PPO	45.32	52.50	94.52	49.80	20.19	18.59	26.23	91.64	26.20	11.36
	GRPO	53.97	60.51	97.34	61.80	32.22	29.50	38.13	93.33	39.20	17.11
	BNPO	52.47	60.37	96.54	59.00	25.94	31.17	36.26	93.76	38.00	17.65
	Dr-GRPO	51.32	62.49	97.17	62.00	29.01	29.59	36.16	93.54	33.60	16.31
	DPH-F	48.89	56.43	96.77	60.80	27.14	27.64	35.59	93.23	39.00	17.11
	DPH-JS	50.17	57.90	97.30	62.80	31.68	27.54	34.29	93.09	35.20	15.78
R.-Reg.	DPO	48.54	55.78	95.30	42.20	19.65	23.25	30.59	92.75	30.80	15.64
	IPO	48.57	54.53	94.89	41.20	19.79	22.10	29.92	92.33	32.40	16.04
	REBEL	49.12	55.34	95.59	50.40	20.19	25.52	31.67	93.01	34.40	16.31
P.-Reg.	MCPO	**56.10**	**71.22**	**99.02**	**88.00**	**59.22**	**57.02**	**50.35**	**95.41**	**48.20**	**21.26**

6.2 Experimental Results

Preference Alignment Results. The preference alignment results are shown in Table 1, where MCPO outperforms all evaluated baselines across every metric. In particular, on highly discriminative benchmarks such as Arena-Hard, the improvement of MCPO over the base model is more than twice that achieved by the baselines, demonstrating that MCPO delivers the most substantial training gains under the same number of iterations. In addition, reward-space regression methods generally underperform state-of-the-art methods based on standard RL dynamics. One possible explanation is that these methods primarily train the model to predict rewards more accurately, rather than directly encouraging it to generate high-reward actions. As a result, their performance gains may be smaller than those of standard RL methods that directly optimize the expected reward.

RLVR Results. The RLVR results are shown in Table 2. MCPO outperforms all evaluated baselines across all metrics. On the relatively easier mathematical reasoning tasks, MCPO achieves an accuracy approximately 4 percentage points higher than the strongest baseline, DPH-F, demonstrating its high training efficiency. On the more challenging coding tasks, MCPO not only achieves the best pass@1 performance, but also exhibits an increasingly larger advantage as the number of attempts increases. Specifically, MCPO outperforms Dr-GRPO by only 1.27 points on pass@1, whereas the gap increases to 3.23 points on pass@5. This suggests that, while approaching the optimal policy more rapidly, MCPO may also avoid regions associated with entropy collapse by following a relatively more direct update trajectory, thereby helping preserve diversity and exploration capability. Meanwhile, although the reward-space regression method A*PO underperforms most

Table 2: RLVR experimental results using Qwen2.5-7B as the base model. The metric pass@k denotes the proportion of problems for which at least one correct solution is generated out of k attempts under identical sampling conditions. **Bold** indicates the best performance.

Method	GSM8k	CodeR1		
	pass@1 ↑	pass@1 ↑	pass@3 ↑	pass@5 ↑
Base	62.44	4.92	5.76	6.18
Reinforce++	79.38	11.52	13.90	14.47
GRPO	77.41	26.97	31.18	32.72
Dr-GRPO	77.63	31.60	38.20	40.45
DPH-F	80.74	17.56	20.93	21.63
DPH-JS	75.36	22.19	25.98	27.39
A*PO	73.92	22.05	26.69	29.63
MCPO	**84.23**	**32.87**	**40.59**	**43.68**

Table 3: Experimental results on Offline RL. We report aggregated scores across three capabilities: Math (accuracy on GSM8k), Reasoning (average score of ARC-E, ARC-C, and PIQA), and Knowledge (average score of MMLU and GPQA). Average denotes the mean score across all individual benchmarks. **Bold** indicates the best performance.

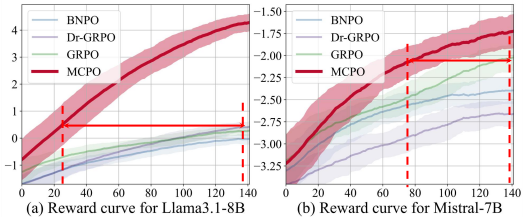
Method	Gemma-7b-it				Llama2-7b-chat-hf			
	Math ↑	Reason. ↑	Know. ↑	Avg. ↑	Math ↑	Reason. ↑	Know. ↑	Avg. ↑
Base	37.07	72.98	28.93	52.31	23.05	44.31	36.00	38.00
DPO	36.62	73.03	30.31	52.72	23.28	44.43	36.22	38.17
IPO	37.07	73.27	29.31	52.58	24.03	44.61	36.73	38.55
SLiC	37.07	72.89	29.13	52.33	23.58	44.41	36.49	38.30
RPO	37.15	73.15	29.18	52.50	23.20	44.28	37.22	38.42
BCO	36.01	73.17	29.06	52.28	23.58	44.43	36.48	38.31
simPO	**38.44**	73.19	28.41	52.47	23.12	44.44	**37.23**	38.48
MCPO	37.60	**73.47**	**30.34**	**53.11**	**24.11**	**44.66**	**37.23**	**38.76**

RL-dynamics-based methods on pass@1, it achieves relatively strong pass@5 performance, suggesting that reward-space regression may also help enhance output diversity.

Offline RL Results. As presented in Table 3, MCPO demonstrates consistent superiority, achieving the highest average scores on both models (53.11 on Gemma and 38.76 on Llama2). In terms of specific capabilities, MCPO excels in logic-intensive tasks, securing the best performance in Math on Llama2 (24.11) and achieving state-of-the-art results in Reasoning across both models (73.47 and 44.66), significantly surpassing strong baselines like IPO and SimPO. Crucially, regarding Knowledge retention, unlike ranking-based methods (e.g., SimPO) which often suffer from performance degradation, MCPO effectively mitigates catastrophic forgetting, achieving the highest Knowledge score on Gemma (30.34) and tying for the best on Llama2 (37.23).

Preference Alignment across Model Scales. Table 4 reports the performance of MCPO on the Alpaca Eval benchmark using Qwen2.5 models of varying sizes (1.5B and 3B). MCPO demonstrates dominant performance across all metrics and model scales, consistently surpassing both the base model and advanced baselines like BNPO and Dr-GRPO. Specifically, on the 1.5B model, MCPO achieves a remarkable Length-Controlled Win Rate (LC_2) of 21.38%, outperforming the second-best baseline. This superiority extends to the 3B model, where MCPO attains the highest scores across all metrics, including a WR_1 of 94.54% and an LC_2 of 50.01%. These results confirm that MCPO's effectiveness is not limited to specific architectures but scales robustly with model size.

Training Speed and Stability. To verify the superior training speed of MCPO, we visualize the reward trajectories during training, as shown in Figure 3. MCPO demonstrates a significantly faster rate of reward growth on both Llama and Mistral, implying that fewer iterations are required to reach a policy of comparable quality. Specifically, on Mistral-7B, MCPO achieves the peak reward attained by the best baseline at the end of one epoch using only approximately 50% of the training steps. On Llama-3-8B, MCPO reaches the same performance milestone with even fewer iterations, further highlighting its exceptional efficiency.



(a) Reward curve for Llama3.1-8B (b) Reward curve for Mistral-7B

Figure 3: The relation between training steps (x-axis) and reward (y-axis) under various algorithms.

We also evaluate the AlpacaEval scores of MCPO models at the corresponding checkpoints, obtaining 53.94, 65.83, and 97.84 on Llama-3-8B, and 30.81, 40.13, and 94.12 on Mistral-7B, which are comparable to the scores achieved by GRPO after a full epoch of training. We note that the convergence advantage on Mistral is slightly less pronounced than on Llama, which we attribute to differences in their initial policies. Since our discussion of the convergence-rate advantage of MCPO

Table 4: Preference alignment results on Alpaca Eval across models of varying parameter sizes. WR_1 and WR_2 denote the Win Rates on versions 1.0 and 2.0, respectively, while LC_2 refers to the Length-Controlled Win Rate on version 2.0. **Bold** indicates the best performance.

Method	Qwen2.5-1.5B-Instruct			Qwen2.5-3B-Instruct		
	$WR_2 \uparrow$	$LC_2 \uparrow$	$WR_1 \uparrow$	$WR_2 \uparrow$	$LC_2 \uparrow$	$WR_1 \uparrow$
Base	8.16	8.41	55.94	35.78	34.12	87.24
GRPO	17.00	12.62	66.69	49.10	44.84	88.74
BNPO	22.53	16.61	72.42	51.70	46.18	91.04
Dr-GRPO	22.76	17.26	72.85	50.41	44.41	90.60
MCPO	**28.79**	**21.38**	**78.32**	**55.79**	**50.01**	**94.54**

is based on tabular softmax policies, this observation may suggest that the Llama policy behaves more similarly to the tabular policy. Moreover, we measure the coefficient of variation of gradient norms during online RLHF training and find that MCPO substantially outperforms GRPO, with values of 0.16 and 0.58, respectively, indicating substantially greater training stability. We attribute this robustness to the use of clipping and baseline techniques.

6.3 Ablation Studies.

We examined the impact of key hyperparameters on MCPO's performance, specifically the clipping thresholds (τ_l, τ_u) and the regularization coefficient β, with results presented in Table 5. We first varied the clipping thresholds magnitudes from 0.1 to 10. The empirical results indicate a non-monotonic relationship: performance does not simply improve with strictly wider or narrower clipping ranges. We attribute this to a trade-off: an excessively wide clipping range exacerbates the detrimental effects of high variance in importance sampling weights, whereas an overly restrictive range mitigates variance but discards critical gradient information, leading to suboptimal learning. We found that asymmetric thresholds of $\tau_l = -0.28, \tau_u = 0.3$ yield the best performance, achieving an optimal balance between variance reduction and information retention. Regarding β, we observed that an overly large coefficient (e.g., $\beta = 0.1$) imposes excessive constraints on the policy update, limiting improvement. Conversely, while a very small coefficient (e.g., $\beta = 0.001$) leads to higher training rewards (approximately 1.0-2.0 points higher than $\beta = 0.01$ finally), it induces severe overfitting, resulting in a slight degradation in Alpaca Eval scores.

Table 5: Ablation studies on Llama3.1-8B-Instruct regarding the clip parameter (τ_l, τ_u) and regularization coefficient β. WR_2 and LC_2 denote the Win Rates and Length-Controlled Win Rate on Alpaca Eval 2.0, respectively. **Bold** indicates the best performance.

Parameters			Alpaca Eval	
β	τ_l	τ_u	$WR_2 \uparrow$	$LC_2 \uparrow$
0.01	-0.28	0.30	**56.10**	**71.22**
0.01	-0.10	0.10	53.36	67.14
0.01	-1.00	1.00	55.93	70.24
0.01	-1.00	5.00	51.48	65.73
0.01	-1.00	10.00	47.01	60.21
0.1	-0.28	0.30	46.36	56.28
0.001	-0.28	0.30	55.23	70.65

7 Conclusion

In this paper, we proposed Minimum Cross-Entropy Policy Optimization (MCPO), a framework that accelerates RL efficiency by directly regressing the policy toward the reparameterized optimal target. MCPO guarantees that the optimal policy is a stationary point of its objective, thereby preserving the optimality of the optimal policy from an optimization perspective. Moreover, analysis under the idealized tabular setting shows that MCPO not only makes monotonic progress toward the optimum, but also, when clipping is inactive, yields the policy closest to the optimal policy among all finite-step updates. By combining the stability of supervised learning with the optimality of RL, MCPO establishes a robust and efficient paradigm for LLM post-training.

AI USE STATEMENT

In this work, we used generative AI tools solely for translation and grammatical editing. They were not used for any core research activities, including theoretical modeling, mathematical formulation, experimental design, method implementation, data processing, or manuscript drafting. Tasks involving synthetic dataset generation or qualitative data analysis are not applicable to this work. We carefully reviewed all AI-assisted content for correctness and take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide sufficient details to facilitate reproduction of our theoretical and empirical results. All major theoretical results are presented in the main text following the derivation flow, with complete proofs provided in Appendix C. The main experimental results and training configurations are reported in the experimental section, while additional implementation details are provided in Appendix B. We use the open-source KW-R1 (Liang et al., 2025) and TRL (von Werra et al., 2020) reinforcement learning frameworks, and all models and datasets used in our experiments are publicly available. Our code will be released upon publication.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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 (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Kianté Brantley, Mingyu Chen, Zhaolin Gao, Jason D Lee, Wen Sun, Wenhao Zhan, and Xuezhou Zhang. Accelerating RL for LLM reasoning with optimal advantage regression. *arXiv preprint arXiv:2505.20686*, 2025.

Peter Chen, Xiaopeng Li, Ziniu Li, Wotao Yin, Xi Chen, and Tianyi Lin. Exploration vs exploitation: Rethinking RLVR through clipping, entropy, and spurious reward. In *International Conference on Learning Representations*, volume 2026, pp. 134506–134540, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting language models with high-quality feedback, 2023.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Zhaolin Gao, Jonathan Chang, Wenhao Zhan, Owen Oertell, Gokul Swamy, Kianté Brantley, Thorsten Joachims, Drew Bagnell, Jason D Lee, and Wen Sun. REBEL: Reinforcement learning via regressing relative rewards. *Advances in Neural Information Processing Systems*, 37:52354–52400, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Jiaqi Han, Mingjian Jiang, Yuxuan Song, Stefano Ermon, and Minkai Xu. f-PO: Generalizing preference optimization with f-divergence minimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 1144–1152. PMLR, 2025.

Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, 2024.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Seungjae Jung, Gunsoo Han, Daniel Wontae Nam, and Kyoung-Woon On. Binary classifier optimization for large language model alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1858–1872, 2025.

Saeed Khaki, JinJin Li, Lan Ma, Liu Yang, and Prathap Ramachandra. RS-DPO: A hybrid rejection sampling and direct preference optimization method for alignment of large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1665–1680, 2024.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, et al. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. In *International Conference on Learning Representations*, volume 2026, pp. 64742–64768, 2026.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023a.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. ReMax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023b.

Jiaqing Liang, Jinyi Han, Xinyi Wang, Zishang Jiang, Chengyuan Xiong, Boyu Zhu, Jie Shi, Weijia Li, Tingyun Li, and Yanghua Xiao. KW-R1: A simple implementation of the GRPO algorithm. https://github.com/lsdefine/simple_GRPO, 2025.

Jiawei Liu and Lingming Zhang. Code-R1: Reproducing R1 for code with reliable rewards. *arXiv preprint arXiv:2503.18470*, 3, 2025.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. In *International conference on learning representations*, volume 2024, pp. 54605–54624, 2024.

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. *arXiv preprint arXiv:2503.20783*, 2025.

Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

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

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.

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.

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.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. KIMI K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

ModelScope Team. EvalScope: Evaluation framework for large models, 2024. URL https://github.com/modelscope/evalscope.

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

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Changyi Xiao, Mengdi Zhang, and Yixin Cao. BNPO: Beta normalization policy optimization. *arXiv preprint arXiv:2506.02864*, 2025.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for LLMs. *Advances in Neural Information Processing Systems*, 37:62279–62309, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

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.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LLAMAFACTORY: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024.

A Theoretical Analysis of Limitations in Traditional RL

A.1 Theoretical Explanation of PPO-based RL Instability

Methods like PPO and GRPO dynamically sample to estimate the advantage A for each token and the objective function $\mathcal{L}$, then unbiasedly maximize $\mathcal{L}$ by maximizing the following objective function $\mathcal{L}_{\text{PPO}}$:

$$\begin{aligned} \mathcal{L}_{\text{PPO}} &= \mathbb{E}_{\pi_{\text{old}}}\left[\min\left\{A\,\omega, A\,\text{clip}\left(\omega, \tau_1, \tau_2\right)\right\}\right] - \beta\,\mathbb{D}_{\text{KL}}\left(\pi||\pi_{\text{ref}}\right) \\ \omega &= \frac{\pi(a|s)}{\pi_{\text{old}}(a|s)}, \quad A = r - \mathbb{E}_{\pi}\left[r\right] \end{aligned} \tag{9}$$

where τ_1 and τ_2 are clipping thresholds, typically set to 0.8 and 1.2.

As can be seen from Equation 9, PPO-based methods utilize the importance sampling ratio ω, which are compelled to employ a clipping mechanism to guarantee convergence (Schulman et al., 2017). Although $\mathcal{L}_{\text{PPO}}$ ensures an unbiased optimization of $\mathcal{L}$, it introduces significant instability in convergence. By taking the derivative of $\mathcal{L}_{\text{PPO}}$, we obtain:

$$\frac{\partial}{\partial\pi(a|s)}\mathcal{L}_{\text{PPO}} \propto \mathbb{I}_c A - \beta\,\frac{\partial}{\partial\pi(a|s)}\mathbb{D}_{\text{KL}}\left(\pi||\pi_{\text{ref}}\right), \tag{10}$$

where $\mathbb{I}_c$ equals 0 when clipping is applied, and 1 otherwise. It can be observed that, the gradient direction is not only determined by the advantage A but also by the rollout policy π_{old}. This means that whenever there is a significant policy update, the gradient term guided by the reward is truncated to 0, causing the model to update the policy in a possibly incorrect direction. This can lead to significant instability in LLM training. The LLM policy π may unexpectedly shift direction due to clipping at any time, and erroneous updates can cause abrupt changes in the advantage distribution, triggering a new cycle of clipping.

A.2 Theoretical Explanation of DPO-based RL bias

DPO (Rafailov et al., 2023) and its variants (Ethayarajh et al., 2024; Hong et al., 2024; Meng et al., 2024; Azar et al., 2024; Han et al., 2025; Khaki et al., 2024) derive the optimality condition from Equation 1 to express the reward r in terms of the policy π. This expression is then incorporated into the loss function for training the reward model, enabling direct policy training through supervised learning of rewards. These approaches achieve the goal of replacing traditional RL policy training with more stable supervised methods and can be uniformly expressed (Han et al., 2025) as $\mathcal{L}_{\text{DPO}}$:

$$\mathcal{L}_{\text{DPO}} = \mathbb{E}_{\pi_{\text{ref}}, r_1 > r_2}\left[f\sigma\left(g_\pi(a_1, s) - g_\pi(a_2, s)\right)\right] \tag{11}$$

where g_π is a monotonically increasing function of π, and f is typically a monotonically increasing convex function.

Although this supervised approach ensures stability, its unbiasedness requires stringent conditions such as an infinite number of samples (Han et al., 2025) and sampling under the optimal policy (Liu et al., 2024). An intuitive way to understand this bias is to examine its gradient with respect to the positive sample policy when each sample appears only once in the dataset:

$$\frac{\partial}{\partial\pi(a_1|s)}\mathcal{L}_{\text{DPO}} \propto \nabla f\sigma\left(g_\pi(a_1, s) - g_\pi(a_2, s)\right)\frac{\partial}{\partial\pi(a_1|s)}g_\pi(a_1, s) \tag{12}$$

Algorithm 1 MCPO for LLMs

Input: Prompt dataset $\mathcal{D}$, reward model r_ϕ, group size G, base model π_θ.
Initialize: $\pi_{\text{old}} \leftarrow \pi_\theta, \pi_{\text{ref}} \leftarrow \pi_\theta$.
repeat
  for each prompt batch $\mathcal{B} \subset \mathcal{D}$ **do**
    Generate G responses for each prompt $s \in \mathcal{B}$ using π_{old}
    Compute rewards $r_\phi(a_i, s)$ for each response
    Construct estimated optimal target $\hat{\pi}^*$ via Equation 5 using π_{ref} and rewards
    Estimate MCPO loss $\mathcal{L}_{\text{MCPO}}(\pi_\theta)$ via Equation 8
    Update π_θ by minimizing $\mathcal{L}_{\text{MCPO}}$ (e.g., via AdamW)
    Update rollout policy $\pi_{\text{old}} \leftarrow \pi_\theta$
  end for
until convergence
Output: Optimized LLM π_θ

From the properties of monotonically increasing functions, it can be observed that this gradient is always greater than zero. Therefore, algorithms like DPO exhibit a strong tendency to overfit, which drives the probability of positive samples to continuously increase toward 1.

B Implementation for LLMs.

We implement MCPO within the mainstream reinforcement learning paradigm (Shao et al., 2024) for LLMs, as outlined in Algorithm 1. Specifically, we maintain three separate LLMs to represent the policies involved: the policy to optimize π_θ, the reference policy π_{ref}, and the rollout (behavior) policy π_{old}. All three models are initialized from the same base model to ensure a consistent starting point.

In each policy update step, we first sample a batch of prompts from the dataset. For each prompt, we generate G candidate responses using the rollout policy π_{old} and evaluate them using the reward model r_ϕ to obtain scalar reward signals. Subsequently, leveraging these rewards and the reference policy π_{ref}, we construct the estimated optimal target $\hat{\pi}^*$. To apply the MCPO objective, we map the text generation process to the standard MDP formulation: the current prompt serves as the state, and the generated response serves as the action. We then estimate the MCPO objective (Equation 8) using the same batch of generated samples. Finally, the parameters of π_θ are updated to minimize this objective. To maintain effective online sampling, we synchronize the rollout policy with the current optimized policy at the end of each iteration.

C Proofs

Theorem 1 Under bounded rewards and $\beta > 0$, the optimal policy π^* is a stationary point of Equation 8.

Proof. Since $\mathbb{E}_\pi[\nabla_\theta \log \pi(a|s)] = 0$ holds for any policy model and parameters θ, we directly analyze the gradient of the MCPO objective:

$$\nabla_\theta \mathcal{L}_{\text{MCPO}} = -\mathbb{E}_\pi \left[\rho_s \, \text{clip} \left\{ \frac{\pi^*(a|s)}{\pi(a|s)}, \frac{\tau_l + 1}{\rho_s}, \frac{\tau_u + 1}{\rho_s} \right\} \nabla_\theta \log \pi(a|s) \right], \tag{13}$$

where $\rho_s = \frac{Z(s)}{\widehat{Z}(s)}$ denotes the estimation error of the partition function.

Substituting $\pi = \pi^*$ into the above expression yields

$$\frac{\pi^*(a|s)}{\pi(a|s)} = 1$$

for all actions a under the same state s. Therefore,

$$\text{clip} \left\{ \frac{\pi^*(a|s)}{\pi(a|s)}, \frac{\tau_l + 1}{\rho_s}, \frac{\tau_u + 1}{\rho_s} \right\}$$

reduces to a state-dependent constant that is independent of a. The gradient then becomes

$$\nabla_\theta \mathcal{L}_{\text{MCPO}} = -c(s)\, \mathbb{E}_{\pi^*}[\nabla_\theta \log \pi^*(a|s)] = 0,$$

where $c(s)$ is the resulting constant for state s. Hence, π^* is a stationary point of Equation 8. □

Lemma 2. *Let π and π^* be two probability distributions. Let $\mu_n(\pi)$ denote the n-th central moment of the random variable $X(a) = \left(\frac{\pi^*(a)}{\pi(a)} - 1\right)$ under the distribution π and $\upsilon_n(\pi^*) = \mathbb{E}_{\pi^*}[X(a)^n]$. Then, the following identity holds for any $n \geq 1$: $\mu_n(\pi) = \sum_{i=1}^{n-1}(-1)^{i+1}\upsilon_{n-i}(\pi^*)$.*

Proof. Clearly, $\mathbb{E}_\pi[X(a)] = 0$, thus:

$$\mu_n(\pi) = \mathbb{E}_\pi\left[(X(a) - \mathbb{E}_\pi[X(a)])^n\right] = \mathbb{E}_\pi[X(a)^n] \tag{14}$$

Substituting $X(a) = \frac{\pi^*(a)}{\pi(a)} - 1$, we obtain:

$$\begin{aligned}
\mu_n(\pi) &= \mathbb{E}_\pi\left[\textstyle\sum_{i=0}^{n}\binom{n}{i}(-1)^i\left(\frac{\pi^*(a)}{\pi(a)}\right)^{n-i}\right] \\
&= \mathbb{E}_\pi\left[\textstyle\sum_{i=0}^{n}\left(\binom{n-1}{i} + \binom{n-1}{i-1}\right)(-1)^i\left(\frac{\pi^*(a)}{\pi(a)}\right)^{n-i}\right] \\
&= \mathbb{E}_\pi\left[\textstyle\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i\left(\frac{\pi^*(a)}{\pi(a)}\right)^{n-i}\right] - \mathbb{E}_\pi\left[X(a)^{n-1}\right] \\
&= \mathbb{E}_{\pi^*}\left[\textstyle\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i\left(\frac{\pi^*(a)}{\pi(a)}\right)^{n-1-i}\right] - \mathbb{E}_\pi\left[X(a)^{n-1}\right] \\
&= \upsilon_{n-1}(\pi^*) - \mathbb{E}_\pi\left[X(a)^{n-1}\right]
\end{aligned} \tag{15}$$

The proof can be completed by using mathematical induction. □

Lemma 3 (Convergence of tabular MCPO without clipping)**.** *Assume bounded rewards, the absence of clipping, and that the density ratio $\frac{\pi^*(a)}{\pi_k(a)}$ is bounded for all iterations k and actions a. Let $\hat{\pi}^*$ denote the estimated optimal policy utilized as the regression target. Provided that the sequence of step sizes $\{\eta_k\}_{k\geq 0}$, as defined in (Cui et al., 2025), satisfies the standard stochastic approximation conditions ($\sum_k \eta_k = \infty, \sum_k \eta_k^2 < \infty$), the sequence of policies $\{\pi_k\}$ generated by MCPO converges to the true optimal policy π^*.*

Proof. Our analysis is based on the iterative form under the softmax policy (Cui et al., 2025). For the convenience of derivation, we omit the state s. Since the rewards are bounded, the ratio ρ in Equation 13 must also be bounded, thus we have $\hat{\pi}_k^*(a) = \rho_k \pi^*(a)$ for any a, where $\hat{\pi}_k^*(a)$ is the estimated optimal policy in the k-th iteration. Let the gradient of an objective function $\hat{\mathcal{L}}$ with respect to the model parameters θ be expressed as:

$$\nabla_\theta \hat{\mathcal{L}} = \mathbb{E}_{\pi_k}[g_k(a)\nabla_\theta \log \pi_k(a)] \tag{16}$$

Then, when the policy update step size at the k-th iteration is η_k, the expression for the newly updated policy π_{k+1} is (Chen et al., 2026; Cui et al., 2025):

$$\pi_{k+1}(a) = \frac{\pi_k(a)\exp(\eta_k g_k(a))}{\mathbb{E}_{\pi_k}[\exp(\eta_k g_k(\cdot))]} \tag{17}$$

Taking the derivatives, we can obtain that the gradient g_k of MCPO is:

$$g_k(a) = \rho_k \frac{\pi^*(a)}{\pi_k(a)} - 1 \tag{18}$$

Thus we have:

$$\begin{aligned}
\pi_{k+1}(a) &= \frac{\pi_k(a)\exp\left(\eta_k\left(\rho_k\frac{\pi^*(a)}{\pi_k(a)} - 1\right)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k\left(\rho_k\frac{\pi^*(\cdot)}{\pi_k(\cdot)} - 1\right)\right)\right]} = \frac{\pi_k(a)\exp\left(\eta_k\left(\rho_k\frac{\pi^*(a)}{\pi_k(a)} - 1\right)\right)\exp(\eta_k - \eta_k\rho_k)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k\left(\rho_k\frac{\pi^*(\cdot)}{\pi_k(\cdot)} - 1\right)\right)\right]\exp(\eta_k - \eta_k\rho_k)} \\
&= \frac{\pi_k(a)\exp\left(\eta_k\rho_k\left(\frac{\pi^*(a)}{\pi_k(a)} - 1\right)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k\rho_k\left(\frac{\pi^*(a)}{\pi_k(a)} - 1\right)\right)\right]}
\end{aligned} \tag{19}$$

We now compute the expression for $\Delta(\pi_{k+1})$ under Equation 17:

$$\begin{aligned}\Delta(\pi_{k+1}) &= \mathbb{D}_{KL}(\pi^*||\pi_k) - \mathbb{D}_{KL}(\pi^*||\pi_{k+1}) \\ &= \mathbb{E}_{\pi^*}\left[\log \frac{\pi_{k+1}(a)}{\pi_k(a)}\right] \\ &= \mathbb{E}_{\pi^*}\left[\log \frac{\exp\left(\eta_k g_k(a)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k g_k(\cdot)\right)\right]}\right] \\ &= \eta_k \mathbb{E}_{\pi^*}\left[g_k(a)\right] - \log \mathbb{E}_{\pi_k}\left[\exp\left(\eta_k g_k(a)\right)\right]\end{aligned} \tag{20}$$

We perform an infinite-order Taylor expansion of $\log \mathbb{E}_{\pi_k}\left[\exp\left(\eta_k g_k(a)\right)\right]$ at $\eta_k \rho_k = 0$, obtaining:

$$\begin{aligned}\Delta(\pi_{k+1}) &= \eta_k \mathbb{E}_{\pi^*}\left[g_k(a)\right] - \log \mathbb{E}_{\pi_k}\left[\exp\left(\eta_k g_k(a)\right)\right] \\ &= \eta_k \rho_k \mu_1(\pi^*) - \frac{\eta_k^2 \rho_k^2}{2}\mu_2(\pi_k) + \textstyle\sum_{n\geq 3} \frac{\rho_k^n \eta_k^n}{n!}(-1)^{n+1}\mu_n(\pi_k),\end{aligned} \tag{21}$$

where $\mu_n(\pi)$ is the n-th central moment of $\frac{\pi^*(a)}{\pi(a)} - 1$ under the distribution of π. By applying Lemma 2, and let $\tilde{\eta}_k = \rho_k \eta_k$, we can proceed with the derivation:

$$\begin{aligned}\Delta(\pi_{k+1}) &= \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*) + \textstyle\sum_{n\geq 3} \frac{\tilde{\eta}_k^n}{n!}(-1)^{n-1}\sum_{i=1}^{n-1}(-1)^i \upsilon_{n-i}(\pi^*) \\ &= \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*) + \textstyle\sum_{n\geq 1}\upsilon_{2n}(\pi^*)\sum_{i\geq 2n+1}\frac{\tilde{\eta}_k^i}{i!} - \sum_{n\geq 1}\upsilon_{2n+1}(\pi^*)\sum_{i\geq 2n+2}\frac{\tilde{\eta}_k^i}{i!} \\ &\geq \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*) + \textstyle\sum_{n\geq 1}\upsilon_{2n}(\pi^*)\frac{\tilde{\eta}_k^{2n+1}}{(2n+1)!} - \upsilon_{2n+1}(\pi^*)\frac{\tilde{\eta}^{2n+2}}{(2n+2)!}\frac{2n+3}{2n+3-\tilde{\eta}_k} \\ &= \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*) + \textstyle\sum_{n\geq 1}\frac{\tilde{\eta}_k^{2n+1}}{(2n+1)!}\left(\upsilon_{2n}(\pi^*) - \upsilon_{2n+1}(\pi^*)\frac{\tilde{\eta}_k}{2n+2}\frac{2n+3}{2n+3-\tilde{\eta}_k}\right)\end{aligned} \tag{22}$$

Since bounded, we may assume without loss of generality that the upper bound of $\frac{\pi^*(a)}{\pi_k(a)}$ is ζ, which yields:

$$\begin{aligned}\frac{\upsilon_{2n+1}(\pi^*)}{\upsilon_{2n}(\pi^*)} &= \frac{\mathbb{E}_{\pi^*}\left[\left(\frac{\pi^*(a)}{\pi_k(a)} - \upsilon_1(\pi^*)\right)\left(\frac{\pi^*(a)}{\pi_k(a)} - \upsilon_1(\pi^*)\right)^{2n}\right]}{\upsilon_{2n}(\pi^*)} \\ &\leq \frac{\zeta \upsilon_{2n}(\pi^*)}{\upsilon_{2n}(\pi^*)} = \zeta\end{aligned} \tag{23}$$

holds for any $n \geq 1$. Since η_k satisfies condition $\sum_k \eta_k^2 < \infty$, and ρ_k is bound, there must exist a k' such that for all $k > k'$, $\tilde{\eta}_k < \frac{20}{5\zeta+4}$ holds. At this point, for all $n \geq 1$ and $k > k'$, the following holds:

$$\upsilon_{2n}(\pi^*) - \upsilon_{2n+1}(\pi^*)\frac{\tilde{\eta}_k}{2n+2}\frac{2n+3}{2n+3-\tilde{\eta}_k} \geq 0 \tag{24}$$

This ensures that each iteration of MCPO moves closer to the optimal policy:

$$\begin{aligned}\Delta(\pi_{k+1}) &\geq \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*) + \textstyle\sum_{n\geq 1}\frac{\tilde{\eta}_k^{2n+1}}{(2n+1)!}\left(\upsilon_{2n}(\pi^*) - \upsilon_{2n+1}(\pi^*)\frac{\tilde{\eta}_k}{2n+2}\frac{2n+3}{2n+3-\tilde{\eta}_k}\right) \\ &\geq \left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*)\end{aligned} \tag{25}$$

Suppose the sequence $\{\pi_k\}$ converges to a non-optimal policy point π'. Clearly, $\pi_{k+1} = \pi_k$ holds for any $\pi_k = \pi^*$, thus there must be $\pi_k \neq \pi^*$ for any k. Then, starting from iteration k', the KL divergence between π' and π^* is:

$$\begin{aligned}\mathbb{D}_{KL}(\pi^*||\pi') &= \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \textstyle\sum_{k>k'}\Delta(\pi_{k+1}) \\ &\leq \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \textstyle\sum_{k>k'}\left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right)\upsilon_1(\pi^*)\end{aligned} \tag{26}$$

Given that $\upsilon_1(\pi^*) > 0$ holds for any $\pi_k \neq \pi^*$, there exists an constant ϵ satisfying $\upsilon_1(\pi^*) > \epsilon$ for any π_k. Thus, the following holds:

$$\begin{aligned}\mathbb{D}_{KL}(\pi^*||\pi') &\leq \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \epsilon\textstyle\sum_{k>k'}\left(\tilde{\eta}_k - \frac{\tilde{\eta}_k^2}{2}\right) \\ &\leq -\infty\end{aligned} \tag{27}$$

This contradicts the non-negativity property of KL divergence. Therefore, we must have $\pi' = \pi^*$, thus completing the proof. □

Lemma 4 (Convergence of tabular MCPO with clipping)**.** *Assume bounded rewards, tabular softmax policies, and suppose that the density ratio $\frac{\pi^*(a)}{\pi_k(a)}$ is bounded for all iterations k and actions a. Assume further that the clipped weights are not all saturated at the same clipping boundary, i.e., they are not uniformly truncated to τ_l for all actions, nor uniformly truncated to τ_u for all actions. Provided that the sequence of step sizes $\{\eta_k\}_{k\geq 0}$, as defined in (Cui et al., 2025), satisfies the standard stochastic approximation conditions ($\sum_k \eta_k = \infty$ and $\sum_k \eta_k^2 < \infty$), the sequence of policies $\{\pi_k\}_{k\geq 0}$ generated by MCPO converges to the true optimal policy π^*.*

Proof. Our analysis is based on the iterative form under the softmax policy (Cui et al., 2025). For the convenience of derivation, we omit the state s. Since the rewards are bounded, the ratio ρ in Equation 13 must also be bounded, thus we have $\hat{\pi}_k^*(a) = \rho_k \pi^*(a)$ for any a, where $\hat{\pi}_k^*(a)$ is the estimated optimal policy in the k-th iteration. Let the gradient of an objective function $\hat{\mathcal{L}}$ with respect to the model parameters θ be expressed as:

$$\nabla_\theta \hat{\mathcal{L}} = \mathbb{E}_{\pi_k}\left[g_k(a)\nabla_\theta \log \pi_k(a)\right] \tag{28}$$

Then, when the policy update step size at the k-th iteration is η_k, the expression for the newly updated tabular softmax policy π_{k+1} is (Cui et al., 2025; Chen et al., 2026):

$$\pi_{k+1}(a) = \frac{\pi_k(a)\exp\left(\eta_k g_k(a)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k g_k(\cdot)\right)\right]} \tag{29}$$

Taking the derivatives, we can obtain that the gradient g_k of MCPO is:

$$g_k(a) = \text{clip}\left(\rho_k \frac{\pi^*(a)}{\pi_k(a)} - 1, \tau_l, \tau_u\right) = \rho_k \text{clip}\left(\frac{\pi^*(a)}{\pi_k(a)}, \frac{\tau_l + 1}{\rho_k}, \frac{\tau_u + 1}{\rho_k}\right) - 1 \tag{30}$$

Note that the two clipping operations, upper and lower, are performed simultaneously. That is, if the previous clipping does not uniformly saturate at either the upper or lower threshold, the same applies to the subsequent clipping. Let $\tilde{\tau}_u = \frac{\tau_u+1}{\rho_k}, \tilde{\tau}_l = \frac{\tau_l+1}{\rho_k}$ and $\sigma_k(a) = \text{clip}\left(\frac{\pi^*(a)}{\pi_k(a)}, \tilde{\tau}_u, \tilde{\tau}_l\right)$, we have:

$$\pi_{k+1}(a) = \frac{\pi_k(a)\exp\left(\eta_k\left(\rho_k\sigma_k(a) - 1\right)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k\left(\rho_k\sigma_k(\cdot) - 1\right)\right)\right]} = \frac{\pi_k(a)\exp\left(\eta_k\rho_k\sigma_k(a)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta_k\rho_k\sigma_k(\cdot)\right)\right]} \tag{31}$$

Let $\tilde{\eta}_k = \rho_k\eta_k$ and substitute Equation 31 into Equation 29. We can then obtain:

$$\begin{aligned}\Delta(\pi_{k+1}) &= \mathbb{D}_{KL}(\pi^*||\pi_k) - \mathbb{D}_{KL}(\pi^*||\pi_{k+1}) \\ &= \mathbb{E}_{\pi^*}\left[\log\frac{\pi_{k+1}(a)}{\pi_k(a)}\right] \\ &= \mathbb{E}_{\pi^*}\left[\log\frac{\exp\left(\tilde{\eta}_k\sigma_k(a)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\tilde{\eta}_k\sigma_k(\cdot)\right)\right]}\right] \\ &= \tilde{\eta}_k\mathbb{E}_{\pi^*}\left[\sigma_k(a)\right] - \log\mathbb{E}_{\pi_k}\left[\exp\left(\tilde{\eta}_k\sigma_k(a)\right)\right]\end{aligned} \tag{32}$$

By performing an Taylor expansion of $\log\mathbb{E}_{\pi_k}\left[\exp\left(\tilde{\eta}_k\sigma_k(a)\right)\right]$ at $\tilde{\eta}_k = 0$, we obtain:

$$\begin{aligned}\Delta(\pi_{k+1}) &= \tilde{\eta}_k\mathbb{E}_{\pi^*}\left[\sigma_k(a)\right] - \log\mathbb{E}_{\pi_k}\left[\exp\left(\tilde{\eta}_k\sigma_k(a)\right)\right] \\ &= \tilde{\eta}_k\left(\mathbb{E}_{\pi^*}\left[\sigma_k\right] - \mathbb{E}_{\pi_k}\left[\sigma_k\right]\right) - \frac{\tilde{\eta}^2}{2}\tilde{\mu}_2(\pi_{k'}),\end{aligned} \tag{33}$$

where $\tilde{\mu}_n(\pi_k)$ is the n-th central moment of σ_k under the distribution of π_k, and $\pi_{k'} = \frac{\pi_k(a)\exp\left(\eta'_k\sigma_k(a)\right)}{\mathbb{E}_{\pi_k}\left[\exp\left(\eta'_k\sigma_k(\cdot)\right)\right]}$ where η' is some constant satisfying $0 \leq \eta' \leq \tilde{\eta}$. Clearly, σ_k is bounded, thus $\tilde{\mu}_2(\pi_{k'}) < \infty$. Given that $\mathbb{E}_{\pi_k}\left[\frac{\pi^*(\cdot)}{\pi_k(\cdot)}\right] = 1$, now we can consider the magnitude of $(\mathbb{E}_{\pi^*}[\sigma_k] - \mathbb{E}_{\pi_k}[\sigma_k])$:

$$\begin{aligned}\mathbb{E}_{\pi^*}[\sigma_k] - \mathbb{E}_{\pi_k}[\sigma_k] &= \mathbb{E}_{\pi_k}\left[\frac{\pi^*(\cdot)}{\pi_k(\cdot)}\sigma_k\right] - \mathbb{E}_{\pi_k}\left[\frac{\pi^*(\cdot)}{\pi_k(\cdot)}\right]\mathbb{E}_{\pi_k}[\sigma_k] \\ &= \mathrm{Cov}_{\pi_k}\left(\frac{\pi^*}{\pi_k}, \sigma_k\right)\end{aligned} \tag{34}$$

Since the clipping mechanism does not uniformly saturate at either the upper or lower thresholds for all actions, σ_k is a monotonic non-decreasing function of the random variable $\frac{\pi^*}{\pi_k}$. Therefore, we have:

$$(\tilde{\mu}_1(\pi^*) - \tilde{\mu}_1(\pi_k)) = \mathrm{Cov}_{\pi_k}\left(\frac{\pi^*}{\pi_k}, \sigma_k\right) > 0 \tag{35}$$

holds for any $\pi_k \neq \pi^*$. Since the rewards are bounded, $\{\rho_k\}$ is also bounded. Therefore, $\{\eta_k\}$ still satisfies the condition:

$$\textstyle\sum_k \tilde{\eta}_k = \infty, \sum_k \tilde{\eta}_k^2 < \infty \tag{36}$$

Clearly, $\pi_{k+1} = \pi_k$ holds for any $\pi_k = \pi^*$, thus any sequence $\{\pi_k\}$ approaching π^* will converge to π^*, and a sequence that does not converge to π^* cannot approach π^* for any k. Assuming $\{\pi_k\}$ eventually converges to $\hat{\pi}$ and $\hat{\pi} \neq \pi^*$, then it must hold that:

$$\begin{aligned}\mathbb{D}_{KL}(\pi^*||\hat{\pi}) &= \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \textstyle\sum_{k>k'} \Delta(\pi_{k+1}) \\ &= \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \textstyle\sum_{k>k'} \tilde{\eta}_k \mathrm{Cov}_{\pi_k}\left(\frac{\pi^*}{\pi_k}, \sigma_k\right) + \sum_{k>k'} \frac{\tilde{\eta}_k^2}{2}\tilde{\mu}_2(\pi_{k'}) \\ &= \mathbb{D}_{KL}(\pi^*||\pi_{k'}) - \infty\end{aligned} \tag{37}$$

Given that $\frac{\pi^*(a)}{\pi_k(a)}$ is bounded for all iterations k and actions a, $\mathbb{D}_{KL}(\pi^*||\pi_{k'})$ is bounded as well. This implies that $\mathbb{D}_{KL}(\pi^*||\hat{\pi}) = -\infty < 0$, which contradicts the non-negativity of KL divergence. Therefore, there must be $\hat{\pi} = \pi^*$. □

Lemma 5. *Let X be a random variable distributed according to π, and let $Z = clip(X, m, n)$ be its truncated version, where $m < n$ are the clipping thresholds. Then, the covariance between X and Z is lower-bounded by the variance: $Cov(X, Z) \geq Var(Z)$.*

Proof. Let $Y = X - Z$, it is clear that Y and Z are both monotonic and non-negatively correlated functions of X. Then there must be:

$$\mathrm{Cov}_\pi(Y, Z) \geq 0 \tag{38}$$

Substituting Y yields the final proof:

$$\mathrm{Cov}_\pi(X, Z) = \mathrm{Cov}_\pi(Y, Z) + \mathrm{Cov}_\pi(Z, Z) \geq \mathrm{Var}_\pi(Z) \tag{39}$$

□

Theorem 2 Assume that the clipped weights are not all saturated at the same clipping boundary, i.e., they are not uniformly truncated to τ_l for all actions, nor uniformly truncated to τ_u for all actions. Let $\eta > 0$ denote the step size, π the current policy, π^* the optimal policy, π_{ref} the reference policy, and r the reward function. Consider the updated tabular softmax policies π_{MCPO} and π_{PPO} obtained after a single iteration using MCPO and the standard PPO-style objective, respectively. Let the one-step improvement toward optimality be defined as $\Delta(\cdot) = \mathbb{D}_{\mathrm{KL}}(\pi^*\|\pi) - \mathbb{D}_{\mathrm{KL}}(\pi^*\|\cdot)$. Then the following properties hold: *(1)* There always exists a step size $\eta > 0$ such that the MCPO update guarantees a strictly positive improvement, i.e., $\Delta(\pi_{\mathrm{MCPO}}) > 0$; *(2)* In contrast, under certain feasible conditions, achieving positive improvement $\Delta(\pi_{\mathrm{PPO}}) > 0$ is impossible regardless of the choice of η.

Proof. Let $\rho = \frac{Z(s)}{\widehat{Z(s)}}$ the estimation error of π^*. Given that $\tilde{\mu}_2(\pi') \leq \mathbb{E}_{\pi'}\left[\text{clip}\left(\frac{\pi^*(\cdot)}{\pi(\cdot)}, \frac{\tau_l+1}{\rho}, \frac{\tau_u+1}{\rho}\right)^2\right] \leq \frac{1}{\rho^2}\max\{(\tau_l+1)^2, (\tau_u+1)^2\}$ holds for any π', substituting the condition into Equation 33, 34 and using Lemma 5, it can be proved that when $\eta < \frac{2\text{Var}_\pi\left(\text{clip}\left(\frac{\hat{\pi}^*(\cdot)}{\pi(\cdot)}-1, \tau_l, \tau_u\right)\right)}{\max\{(\tau_l+1)^2, (\tau_u+1)^2\}}$, inequality (1) always holds. Here, we only detail the substitution process for proving (2). Following the same derivation in Equation 25, the expression for $\Delta(\pi')$ is:

$$\begin{aligned}\Delta(\pi') &= \mathbb{D}_{KL}(\pi^*||\pi) - \mathbb{D}_{KL}(\pi^*||\pi') \\ &= \mathbb{E}_{\pi^*}\left[\log\frac{\pi'(a)}{\pi(a)}\right] \\ &= \mathbb{E}_{\pi^*}\left[\log\frac{\exp\left(\eta g(a)\right)}{\mathbb{E}_\pi\left[\exp\left(\eta g(\cdot)\right)\right]}\right] \\ &= \eta\mathbb{E}_{\pi^*}\left[g(a)\right] - \log\mathbb{E}_\pi\left[\exp\left(\eta g(a)\right)\right]\end{aligned} \tag{40}$$

Taking the derivatives, we can obtain that the gradient g_{RL} of Equation 9 is:

$$g_{RL}(a) = \mathbb{I}_c r(a) - \beta\log\frac{\pi(a)}{\pi_{\text{ref}}(a)}, \tag{41}$$

where $\mathbb{I}_c$ equals 1 when the term is not clipped, and 0 otherwise.

By performing a Taylor expansion on $\log\mathbb{E}_\pi\left[\exp\left(\eta g(a)\right)\right]$ from $\eta = 0$, we can obtain:

$$\log\mathbb{E}_\pi\left[\exp\left(\eta g(a)\right)\right] = \eta\mathbb{E}_\pi\left[g(a)\right] + \frac{\eta^2}{2}\text{Var}_{\tilde{\pi}}\left[g(a)\right], \tag{42}$$

where $\tilde{\pi}(a) = \frac{\pi(a)\exp(\tilde{\eta}g(a))}{\mathbb{E}_\pi[\exp(\tilde{\eta}g(\cdot))]}$ and $\tilde{\eta}$ is a constant satisfying $0 \leq \tilde{\eta} \leq \eta$. Substituting Equation 42 and 41 into Equation 40 yields the formulations of the RL improvements:

$$\Delta(\pi_{RL}) = \eta\mathbb{E}_{\pi^*}\left[\tilde{A}\right] - \frac{\eta^2}{2}\text{Var}_{\tilde{\pi}}\left[\tilde{A}\right], \tag{43}$$

where $\tilde{A} = \mathbb{I}_c r(a) - \mathbb{E}_\pi\left[\mathbb{I}_c r(a)\right] - \beta\log\frac{\pi(a)}{\pi_{\text{ref}}(a)} + \beta\mathbb{D}_{KL}\left(\pi||\pi_{\text{ref}}\right)$. We let $\beta\log\frac{\pi(a)}{\pi_{\text{ref}}(a)} - \beta\frac{\pi^*(a)}{\pi(a)} - \mathbb{I}_c r(a) + \mathbb{E}_\pi\left[\mathbb{I}_c r(a)\right] = \lambda$, where λ is an arbitrary constant independent of the action, which gives:

$$\begin{aligned}\Delta(\pi_{RL}) &= \eta\mathbb{E}_{\pi^*}\left[\tilde{A}\right] - \frac{\eta^2}{2}\text{Var}_{\tilde{\pi}}\left[\tilde{A}\right], \\ &\leq \eta\mathbb{E}_{\pi^*}\left[\tilde{A}\right] \\ &= \eta\left(\textstyle\sum_a \pi^*(a)\left(\mathbb{I}_c r(a) - \mathbb{E}_\pi\left[\mathbb{I}_c r(a)\right] - \beta\log\frac{\pi(a)}{\pi_{\text{ref}}(a)} + \beta\mathbb{D}_{KL}\left(\pi||\pi_{\text{ref}}\right)\right)\right) \\ &= \eta\left(\beta\mathbb{D}_{KL}\left(\pi||\pi_{\text{ref}}\right) - \textstyle\sum_a \pi^*(a)\left(\lambda + \beta\frac{\pi^*(a)}{\pi(a)}\right)\right) \\ &= \eta\left(\textstyle\sum_a \pi(a)\left(\lambda + \beta\frac{\pi^*(a)}{\pi(a)} + \mathbb{I}_c r(a) - \mathbb{E}_\pi\left[\mathbb{I}_c r(a)\right]\right) - \sum_a \pi^*(a)\left(\lambda + \beta\frac{\pi^*(a)}{\pi(a)}\right)\right) \\ &= \eta\beta\left(1 - \mathbb{E}_{\pi^*}\left[\frac{\pi^*(a)}{\pi(a)}\right]\right) \\ &\leq 0\end{aligned} \tag{44}$$

□

Lemma 1. Under a tabular softmax policy optimized with natural policy gradient, for a sufficiently small common step size, using the cross-entropy with either the optimal policy π^* or a reparameterized optimal target $\hat{\pi}^* = \rho\pi^*, \rho > 0$ as the objective maximizes the one-step progress toward π^* among all objective functions, thereby inducing the most direct update toward the optimal policy.

Proof. We first consider the gradient of the cross-entropy objective:

$$-\nabla\mathbb{E}_p\left[\log\pi\right] = \mathbb{E}_\pi\left[\nabla\log\pi(a)\right] - \mathbb{E}_\pi\left[\frac{p(a)}{\pi(a)}\nabla\log\pi(a)\right] = \mathbb{E}_\pi\left[\left(1 - \frac{p(a)}{\pi(a)}\right)\nabla\log\pi(a)\right], \tag{45}$$

where p is an arbitrary nonnegative variable and $\mathbb{E}_\pi[\nabla \log \pi] = 0$.

Using Equations 40 and 42, we can derive the expression for the one-step improvement $\Delta(\pi')$ as follows:

$$\Delta(\pi') = \left(\mathbb{E}_{\pi^*}\left[g(a)\right] - \mathbb{E}_\pi\left[g(a)\right]\right)\eta + O(\eta^2), \tag{46}$$

where $g(a)$ denotes the partial derivative of the objective function with respect to the policy, and varies across algorithms. When the step size η is sufficiently small, we only need to consider the linear term. By further imposing the bounded expected-gradient condition and requiring $g(a)$ to have unit norm, we obtain the following formal objective for maximizing the one-step improvement:

$$\begin{aligned} &\max_g \mathbb{E}_{\pi^*}\left[g(a)\right] - \mathbb{E}_\pi\left[g(a)\right] \\ &s.t. \mathbb{E}_\pi\left[g(a)^2\right] = \tau \end{aligned} \tag{47}$$

The KKT conditions for the optimal solution of this problem are given by:

$$\pi^*(a) - \pi(a) + 2\lambda\pi(a)g(a) \tag{48}$$

Substituting Equation 30 into the above conditions, we can verify that it satisfies this form, thereby completing the proof. This conclusion can be further extended to regression toward $\hat{\pi}^*$:

$$g(a) \propto 1 - \frac{\pi^*(a)}{\pi(a)} = \left(1 + \frac{1}{\rho}\right) + \frac{1}{\rho}\left(1 - \frac{\hat{\pi}^*(a)}{\pi(a)}\right). \tag{49}$$

Since $\mathbb{E}_\pi[\nabla \log \pi(a)] = 0$, we have:

$$\begin{aligned} \mathbb{E}_\pi\left[g(a)\nabla \log \pi(a)\right] &= \rho\, \mathbb{E}_\pi\left[\left(1 - \frac{\hat{\pi}^*(a)}{\pi(a)}\right)\nabla \log \pi(a)\right] + \left(1 + \frac{1}{\rho}\right)\mathbb{E}_\pi\left[\nabla \log \pi(a)\right] \\ &= \rho\, \mathbb{E}_\pi\left[\left(1 - \frac{\hat{\pi}^*(a)}{\pi(a)}\right)\nabla \log \pi(a)\right], \end{aligned} \tag{50}$$

which is identical to the cross-entropy gradient with $\hat{\pi}^*$ as the target. □

D Experiment Details

D.1 Preference Alignment Experiments

General Parameters. To ensure fairness, we made every effort to keep the hyperparameters identical or closely matched across all baseline methods. We allocated 80GB A100 GPU resources for each experiment, consistently using trl version 0.19.1 along with corresponding versions of other dependencies. Among these, PPO requires two A100 GPUs for training because it needs to train both the critics model and the policy model simultaneously, while the other experiments only require one A100 GPU. We applied uniform LoRA fine-tuning to all LLMs to conserve GPU memory, setting the rank to 8 and the alpha value to 16. The maximum response length for the models was set to 512 tokens, which is slightly above the average response length of 7B models and rarely truncates outputs from models of this scale or smaller. For models with 7B parameters and above, the group size is set to 3, while for models with 7B parameters and below, it is set to 4. The batch size for all models is set to 96. Consistent with related work (Rafailov et al., 2023; Li et al., 2023b; Ahmadian et al., 2024), each experiment is trained for 1 epoch. However, since the group size for PPO is effectively 1, PPO is trained for 3 epochs. Under this configuration, training a 7B parameter model and larger requires approximately 120 hours, a 3B model takes about 80 hours, and a 1.5B model needs roughly 45 hours.

PPO. The beta for PPO was set to 0.01, which is the same as in GRPO, and the gamma for advantage calculation was set to 0.95. The critic model was initialized using the reward model, following the official trlexample, and was fine-tuned with LoRA (rank=4, alpha=8) to prevent out-of-memory errors. Other hyperparameters, such as the learning rate and the weights for the critic loss and policy loss, all used the default values from trl.

GRPO, BNPO and Dr-GRPO. We initially conducted three sets of experiments with beta values of 0.1, 0.02, 0.01, 0.005 and 0.001 to compare with GRPO's performance, and ultimately selected 0.01 as the final beta value based on its superior results. The experimental results for different beta values

Table 6: Performance improvement on Llama3.1-8B-Instruct with different betas using GRPO.

beta	AlpacaEval-WR	AlpacaEval-LC
Base Model	42.13	46.72
0.1	50.98	57.22
0.02	53.49	57.96
0.01	**53.97**	**60.51**
0.005	43.25	50.18
0.001	45.05	49.82

are shown in Table 6. Since the objective functions are fundamentally the same, both utilize KL divergence for constraints and only alter the sequence of loss averaging, thus BNPO and Dr-GRPO employ the same beta value as GRPO.

DPO and IPO. We implemented online DPO and online IPO using the $OnlineDPOTrainer$ from trl. The beta value was set to 0.01, consistent with the GRPO configuration. Since the model's performance improvement was limited with the default learning rate of 1e-6, we increased the learning rate to 3e-6. Other hyperparameters, including label smoothing, were set to their default values in trl.

DPH-F, DPH-JS, f-PO. These methods were implemented exactly as described in their original papers, using GRPOTrainer as the base class in trl and overriding the $_compute_loss$ function. For f-PO, we adopted the list-wise objective introduced in the paper. DPH-F and DPH-JS used the same learning rate as GRPO, while f-PO used its best-performing values, namely 1e-5 and for f-PO. All four methods used a beta value of 0.01 to ensure comparability with GRPO.

MCPO. MCPO is implemented by inheriting from the $GRPOTrainer$ class in trl and overriding the $_compute_loss$ method. The clipping thresholds for MCPO are set to -0.28 and 0.3, while other parameters such as the learning rate and beta remain the same as those in GRPO.

Evaluations. We used the official code implementations for all evaluations. For AlpacaEval, we employed Version 2's Win Rate (WR), Length-Controlled Win Rate (LC), and Version 1's Win Rate (v1.0) as metrics. We consistently use a repetition penalty of 1.2, a temperature of 0.7, and a top_p of 0.95 when generating responses with LLM. For Arena-Hard, we used the respective win rates from v0.1 and v2.0 as metrics.

D.2 RLVR Experiment

Codes. All of our code is based on a 1.5k-star RL framework from GitHub, named KW-R1 (Liang et al., 2025). We use exactly the same versions of all packages as KW-R1, including torch, deepspeed, vllm, etc.

Prompts. To guide the LLM to explicitly output the chain of thought and present it in the required format, we used a system prompt similar to DeepSeek-R1's for mathematical tasks (Guo et al., 2025):

```
You are a helpful assistant. 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 and answer are enclosed within <think>
</think> and <answer> </answer> tags, respectively, i.e.,
<think> reasoning process here </think> <answer> answer here
</answer>.
```

Similarly, for code tasks, we also designed a system prompt to induce the model to output a chain of thought.

```
You are a helpful programming assistant. The user will ask you a
question and you as the assistant solve it. The assistant first
thinks how to solve the task through reasoning and then provides
```

```
the user with the final answer. The reasoning process and answer
are enclosed within <think>...</think> and <answer>...</answer>
tags, respectively.
```

The user prompts remained unaltered from the original question texts in the dataset.

Rewards. The reward setting for the mathematical task training is the same as that of KW-R1 (Liang et al., 2025). A correct answer earns 1 point as the result reward, and a correct format earns 0.2 points as the format reward; otherwise is 0. The final reward is the sum of the result reward and the format reward. During code training, we used the exact same reward function as Code-R1 (Liu & Zhang, 2025). If the format is incorrect, the reward is -1.1 regardless of the result. If the format is correct but the result is wrong, the reward is 0. A fully correct response (both format and result) receives a reward of 1.

Training. We conducted training using 8 H100 GPUs. For mathematical tasks, 1 GPU was allocated for rollout, 1 GPU for computing the reference policy, and 6 GPUs for training. For code tasks, 2 GPUs were used for rollout, 1 GPU for computing the reference policy, and 5 GPUs for training. Additionally, 32 processes were spawned to verify code correctness within a sandboxed environment. Due to the relative simplicity of GSM8K, the group size was set to 8, whereas for the more challenging code tasks, the group size was set to 16. The batch size was set to 512 for all tasks, and we performed full-parameter fine-tuning of the LLM. Similarly, the beta value for all baseline methods was set to 0.001. This specific value was determined to be optimal for achieving the best results in GRPO. Training on GSM8K was conducted for 11,200 steps, taking approximately 8 hours, while training for Code-R1 consisted of 22,000 steps and required approximately 20 hours. The learning rate for GSM8K was set to 3e-6, and for Code-R1 it was set to 1e-6, both of which were verified to be optimal in GRPO.

Evaluation. We used the validation sets of GSM8K and Code-R1 for the final testing. On GSM8K, we directly calculated the answer accuracy, disregarding the format. On Code-R1, both factors were considered, and we sequentially calculated the probability of having at least one correct answer in 1, 3, and 5 attempts as the final metric.

D.3 Offline RL Experiment

Training parameters. All our baselines were implemented using the trl, and the optimal beta is verified to be 0.05 on DPO. All training were conducted for only one epoch, requiring approximately 9 hours on a simgle H100 GPU. The maximum response length was set to 784 tokens and LoRA with a rank of 8 and an alpha of 16 is also applied across all experiments. The implementation of MCPO simply inherits from the DPOTrainer class and overrides the loss function.

Evaluation parameters. All tests utilized the default evaluation methods and parameters of their respective libraries, including the sampling temperature, prompt format, and number of few-shot examples.

E Case study

Prompt:

```
How do I dice without slicing my finger?
```

Response from Qwen2.5-3B-Instruct-GRPO (a):

```
Dicing vegetables or any other food item can be an easy task
if you follow the proper techniques and use appropriate tools
to protect your fingers from injury.

Here's how you can safely dice:

### 1. **Choose Appropriate Tools**
- Use **sharp knives** that fit well in your hand.
```

```
- Invest in good-quality kitchen shears for cutting
harder items like potatoes into small pieces (e.g., mandoline slicers).
- Wear **cut-resistant gloves**, especially when
handling raw materials which might have sharp edges or bones.

### 2. **Safety Precautions:**

- **Wear Gloves:** This reduces accidental contact with blades.

- **Keep Knuckles Away:** Always keep your knuckles
away from where the knife blade meets the chopping board while dicing.

- **Use Proper Stance:** Stand straight and stable;
this will help maintain control over the knife.

- **Maintain Consistent Motion:** Move smoothly across the vegetable
rather than pushing
down hard on one spot as it increases pressure against the knife's
edge leading to accidents.

- **Position Safely:** Keep hands at least two inches apart on
either side of the cut line, not directly under the knife.

### 3. **Safe Techniques When Dicing:**

#### For Small Items:
- If using a chef's knife or larger chopper, hold the ingredient
firmly but gently between thumb and forefinger near its base.
- Position the tip of the knife just above the desired point where
cuts should start.
- Using quick, controlled motions along the length of the ingredient,
make even slices through all layers simultaneously.

#### For Larger Slices:
- Similar technique applies { grip lightly yet securely around the
center part.
- Make vertical cuts first then horizontal ones downwards towards
you until fully separated.

### 4. **Alternative Methods:**
If traditional dicing seems too risky, consider alternatives such
as julienne strips, matchsticks, or thin rounds depending on what
is needed based on recipe requirements.

Using these steps and precautions, you'll find yourself able to chop
ingredients efficiently and harmlessly! Remember practice makes
perfect|don't hesitate to try different methods to get comfortable
with each tool. Safe cooking starts here!
```

Response from Qwen2.5-3B-Instruct-MCPO (b):

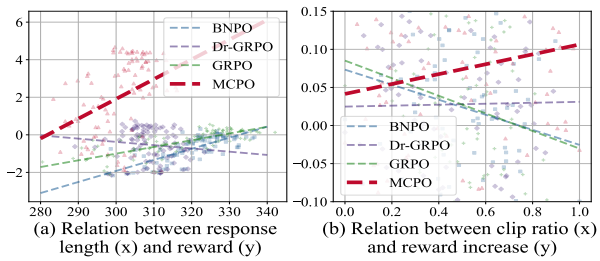
```
Dice-ing ingredients safely and efficiently can be done with the right
techniques to minimize the risk of injury. Here's how you can dicing
effectively while reducing your chances of cutting yourself:

### 1. **Use Proper Tools**
- **Sharp Knife:** Always use a sharp knife for accurate cuts that
require minimal effort.
- **Cutting Board:** A stable and non-slip surface is crucial.
```

```
Ensure it's clean and free from any debris or food particles.

### 2. **Master Your Grip**
- **Safe Hand Position:** Hold the knife firmly but comfortably
in one hand, placing your thumb on top near the blade handle
(this allows better control). Use your other hand to steady the
ingredient being cut, guiding it towards the knife gently rather
than pushing hard against it.

### 3. **Align Ingredients Correctly**
- **Straight Cut Lines:** When chopping, ensure the line along which
you're making the cut is straight. This helps maintain precision and
reduces the need for rapid movements around corners.
- **Even Pressure:** Apply even pressure consistently as you make
each slice. Sudden changes in force increase the likelihood of accidents.

### 4. **Practice Slow, Controlled Movements**
- **Slow Cutting:** Move slowly and deliberately when using knives.
Avoid quick motions; they're harder to stop suddenly if needed.
- **Mindful Awareness:** Stay focused on what you're doing. Be aware
of both hands' positions at all times during the process.

### 5. **Protect Yourself**
- **Gloves:** If available, wearing gloves not only protects your
skin but also adds another layer between you and potential hazards
like slipping or hitting surfaces accidentally.
- **Safety Measures:** Keep fingers away from the part where the
knife meets the board. Never let them rest directly above the point
of contact unless fully supported.

By following these tips, you'll significantly reduce the risks
associated with kitchen knife usage, ensuring safer preparation
processes every time. Remember practice makes perfect|like any skill,
becoming more comfortable and confident will come with experience!
```

Response from Qwen2.5-3B-Instruct-MCPO is better (b). Here is the simplified breakdown: Response (b) is safe but basic. While it misses the most important technique for dicing (the "claw grip"), its advice—use a sharp knife, use a stable board, and move slowly—is correct and helpful. Response (a) is dangerous and confusing. It contains nonsense advice, such as using "kitchen shears" (scissors) to cut potatoes and incorrectly calling them "mandoline slicers." It also suggests keeping your hands "two inches apart," which makes it impossible to hold the food you are trying to dice.

F Global Update Direction and Training Stability.

To verify that MCPO is guided by a global optimal direction and exhibits superior convergence stability, we analyzed the correlations between different training dynamics, as illustrated in Figure 4. In Figure 4(a), we plot the relationship between reward and response length. We observe that baselines like GRPO tend to indiscriminately increase response length with limited reward gains. In contrast, MCPO achieves significant reward improvement with only half the length growth of GRPO. This suggests that tradition methods likely follow a gradient direction towards local optima, where simply extrapolating length is the easiest pattern for LLMs to exploit. Conversely, MCPO, guided by the global optimal direction, effectively bypasses such superficial biases. Figure 4(b) visualizes the relationship between the normalized clipping ratio of a batch and the normalized reward improvement in the subsequent batch ($\mathbb{E}_{\pi_{k+1}}[r] - \mathbb{E}_{\pi_k}[r]$). Theoretically, the subsequent batch should yield higher rewards as the sampling policy has been updated. However, baselines like GRPO exhibit reward degradation (non-positive growth) as the clipping ratio increases, indicating that the clipping mechanism in standard RL architectures, rather than enhancing stability by reducing



(a) Relation between response length (x) and reward (y)

(b) Relation between clip ratio (x) and reward increase (y)

Figure 4: Relationships between different statistical measures during model training.

variance, actually destabilizes the training process. In contrast, MCPO maintains a consistent positive correlation, demonstrating that its clipping mechanism effectively reinforces training stability.

G Broader Impact

This paper proposes a method for post training large language models. Although the algorithm design does not have a direct correspondence to real-world applications, there are still potential societal impacts due to the involvement of LLMs. First, reinforcement learning–based post training of LLMs is widely used to enhance their safety, helpfulness, and bias mitigation, which has positive effects in these areas. However, such methods are also commonly employed in jailbreak attacks, so future work must be careful to ensure proper and responsible use.

H Limitation

Constrained by computational resources, although the theoretical foundation of MCPO does not impose any restrictions on the input-output patterns of the model, currently we have only experimented with text-based autoregressive models. Future work could extend MCPO to multimodal models and other generative models, such as diffusion models.