# ODRPO: ORDINAL DECOMPOSITIONS OF DISCRETE REWARDS FOR ROBUST POLICY OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The alignment of Large Language Models (LLMs) increasingly relies on Reinforcement Learning from AI Feedback (RLAIF) for non-verifiable domains such as long-form question answering and open-ended instruction following. These domains often use LLM-based auto-raters to provide granular, multi-tiered discrete rewards (e.g., 1–10 rubrics) that can be stochastic due to prompt sensitivity and sampling randomness. We consider the black-box setting in which the optimizer observes only the returned discrete reward and does not assume access to the auto-rater's parameters, logits, hidden states, or auxiliary confidence estimates. We empirically characterize this stochasticity across multiple auto-rater configurations. Such variability enters standard advantage estimators such as GRPO and MaxRL through group-normalization statistics, allowing a noisy reward sample to alter the update assigned to the entire group. Repeatedly sampling and aggregating rewards may reduce this noise, but doing so substantially increases evaluation cost. To address this bottleneck, we introduce **O**rdinal **D**ecomposition for **R**obust **P**olicy **O**ptimization (**ODRPO**), a framework that decomposes discrete rewards into a sequence of ordinal binary indicators. By independently computing and accumulating advantages across progressively challenging success thresholds, ODRPO localizes score variability to the thresholds crossed by a score change while supporting flexible weighting schemes and an implicit ordinal curriculum. Empirically, across GRPO, MaxRL, and OTB, three policy models, and seven evaluation benchmarks, all main-experiment configurations ODRPO and its variance-aware variant achieve positive mean relative gains ranging from $2.29\%$ to $6.96\%$. Ablations over auto-rater temperature and reward granularity further show consistent improvements over GRPO, with finer reward resolution producing the clearest trend. These gains incur no material per-step training overhead. ODRPO further attains the noise robustness sought by repeated auto-rater sampling, surpassing MaxRL with 32-sample majority voting on AlpacaEval while using a single reward sample. Finally, our theoretical analysis establishes that the ODRPO formulation retains a well-defined global scalar objective in arbitrary discrete reward spaces when built on estimators with valid binary objectives.

## 1 INTRODUCTION

Reinforcement learning has become a central post-training method for Large Language Models (LLMs). The success of reinforcement learning with verifiable rewards (RLVR) in coding and mathematics (Guo et al., 2025) has motivated its extension to domains such as puzzle solving, scientific literature search, and visual perception (Chen et al., 2025; Burgess et al., 2026; Wang et al., 2025). Many tasks, however, including long-form question answering and open-ended instruction following, lack fixed ground-truth answers and therefore cannot be addressed directly through RLVR. For these tasks, reinforcement learning from AI feedback (RLAIF) substitutes an auto-rater, or "LLM-as-judge," for the verifier, using rubric-based evaluation as a scalable surrogate reward intended to reflect human preferences (Lee et al., 2024; Gunjal et al., 2026). We consider a black-box setting in which the auto-rater returns a discrete score in a multi-tiered reward space through a text interface, while its parameters, logits, hidden states, and auxiliary confidence estimates remain unavailable to the optimizer.



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.

Although this black-box interface makes RLAIF broadly deployable, it requires the optimizer to treat each returned score as a reliable learning signal. In practice, auto-raters are sensitive to sampling randomness, rubric misinterpretation, and position bias (Zhao et al., 2025; Shi et al., 2025; Li et al., 2026a), which can produce noisy or inconsistent rewards. This unreliability is consequential for standard advantage estimators such as GRPO (Shao et al., 2024) and MaxRL (Tajwar et al., 2026), which compute advantages directly from the observed group rewards: a single perturbed score can alter the group-normalization statistics and thereby affect the update assigned to every response in the group.

In this work, we introduce **O**rdinal **D**ecomposition for **R**obust **P**olicy **O**ptimization (ODRPO), a plug-and-play augmentation framework for advantage estimators to improve their robustness against noisy auto-rater signals. Given an ordered discrete reward space, ODRPO represents each observed score as a sequence of binary indicators, one for each success threshold attained by that score. It applies the underlying estimator independently to the binary outcome at each threshold and sums the resulting threshold-level advantages to form the advantage used by the unchanged policy-optimization procedure. Consequently, ODRPO can wrap estimators such as GRPO, MaxRL, and OTB (Li et al., 2026b) without access to the auto-rater's internal state. The decomposition preserves the ordering of reward levels and localizes changes: a change in score modifies only the thresholds it crosses.

We apply ODRPO to GRPO, MaxRL, and OTB and evaluate it across three policy models and seven benchmarks. Across these experiments, all ODRPO and ODRPO-Gini configurations achieve positive mean relative gains, ranging from $2.29\%$ to $6.96\%$, while maintaining per-step runtimes comparable to the corresponding baselines. The decomposition also addresses a theoretical limitation of applying binary-reward estimators directly to multi-tier rewards. In arbitrary discrete reward spaces, the induced update field need not correspond to the gradient of a global scalar objective. We show that ODRPO recovers a well-defined global objective whenever its underlying estimator admits a valid objective for binary rewards: the threshold-wise binary objectives combine into a scalar objective for the original discrete reward space. This result connects the plug-and-play construction to an explicit optimization objective rather than treating it only as an empirical variance-reduction heuristic.

Our contributions are fourfold. First, we empirically characterize auto-rater stochasticity in multi-tiered reward spaces using per-response distribution statistics and ranking concordance across multiple policy–auto-rater–dataset configurations. Second, we introduce ODRPO as a plug-and-play transformation that converts a discrete reward into ordinal binary thresholds and composes with existing advantage estimators. Third, we establish that this construction preserves a well-defined global scalar objective for arbitrary discrete reward spaces whenever the underlying estimator has a valid binary objective. Fourth, we validate the framework across multiple advantage estimators, policy models, auto-raters, and evaluation dimensions without additional sampling or material per-step computational overhead.

## 2 Auto-rater Rewards Exhibit Stochasticity.

RLAIF has become increasingly important and widely used when extending from RLVR to non-verifiable domains. However, evidence from recent studies suggests that LLM-based auto-raters often exhibit high variance due to prompt sensitivity and sampling randomness (Zhao et al., 2025; Shi et al., 2025; Li et al., 2026a). This raises concerns about whether such stochasticity manifests as inconsistent reward signals, which would fundamentally undermine their reliability as stable feedback for RL. Therefore, we provide a systematic examination to analyze the problem.

An auto-rater is characterized as stochastic if $N$ independent evaluations of $M$ responses to a fixed prompt yield frequent rank-flips, manifesting as a lack of consensus and high evaluative variance. Rank agreement can be quantified using Kendall's coefficient of concordance, or Kendall's $W$ (Kendall & Smith, 1939). If $W \sim 0$ then the auto-rater is highly inconsistent and if $W \sim 1$ then the auto-rater is consistent. Because the $N$ "virtual judges" are repeated samples from the same auto-rater, a reliable reward signal should exhibit $W$ close to unity, indicating that response rankings remain stable across evaluations.

We performed the statistical analysis under four policy-auto-rater-dataset configurations. The main text reports the two configurations that use Qwen3-30B-A3B-Instruct (Team, 2025) as the auto-rater on UltraFeedback (Cui et al., 2023), with Qwen2.5-7B-Instruct (Qwen et al., 2025) and Qwen3-

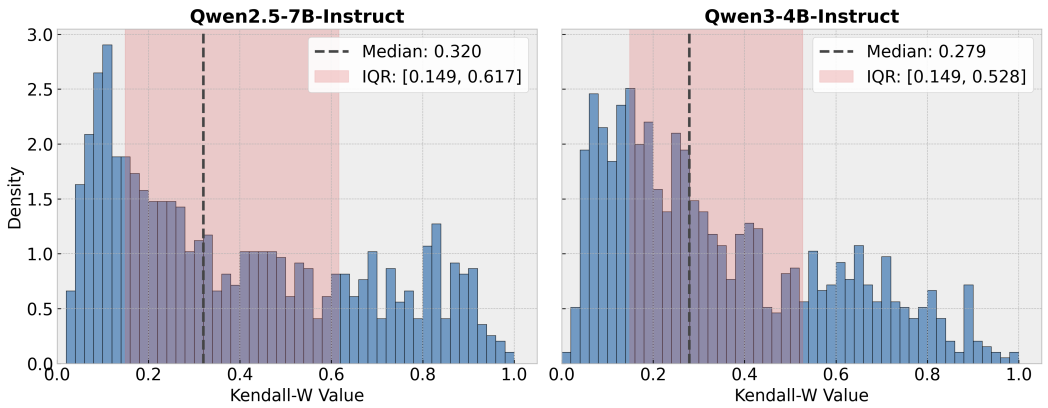



Figure 1: Distribution of Kendall's $W$ (concordance) for the Qwen3-30B-A3B-Instruct auto-rater on the UltraFeedback dataset. Each panel corresponds to one policy model whose $M = 8$ responses per prompt are scored $N = 16$ times, over 1,000 prompts. Median $W$ is 0.320 for Qwen2.5-7B-Instruct and 0.279 for Qwen3-4B-Instruct, with interquartile ranges $[0.149, 0.617]$ and $[0.149, 0.528]$, respectively. Dashed lines indicate medians and shaded regions indicate interquartile ranges. The substantial mass at low $W$ values indicates weak agreement among repeated auto-rater judgments for many prompts under both policies.

4B-Instruct as the policy models. The two remaining configurations use GPT-OSS-120B (OpenAI, 2025) in its low-thinking mode as the auto-rater, paired with Qwen2.5-7B-Instruct on UltraFeedback and with OLMo-2-7B-Instruct (OLMo et al., 2025) on WildChat (Zhao et al., 2024), and are deferred to Appendix section A. For each configuration, we randomly sampled 1,000 prompts from the corresponding dataset and generated $M = 8$ responses per prompt using the policy model. Each response was independently assigned $N = 16$ scores by the auto-rater, with both auto-raters producing scores between $1 - 10$. This procedure yielded 1,000 score matrices of size $M \times N$ per configuration. We computed Kendall's coefficient of concordance ($W$) and its significance test for each matrix using Pingouin (Vallat, 2018). The Kendall's $W$ values for the two Qwen3-30B-A3B-Instruct configurations are shown in fig. 1; the GPT-OSS-120B configurations and further distributional analysis are provided in Appendix section A.

This empirical assessment demonstrates that a high-capacity auto-rater such as Qwen3-30B-A3B-Instruct can produce inconsistent scoring distributions for identical inputs, resulting in a high-variance reward signal. The same pattern holds for GPT-OSS-120B across two policies and two datasets (Appendix section A). This stochasticity propagates through the advantage estimator, potentially corrupting the gradient signal and destabilizing the optimization process. These findings highlight the necessity of a variance-aware advantage estimator, one capable of dissipating auto-rater's noise across ranking levels to localize and mitigate its impact on policy updates.

## 3 ODRPO: Ordinal Decomposition for Robust Policy Optimization

We treat the auto-rater as a black-box stochastic evaluator that maps a prompt $x$ and response $y$ to a discrete reward $r \in \{1, \ldots, K\}$, where $K \geq 2$. The optimizer observes only this reward, without access to evaluator internals or confidence estimates. Under this setting, **O**rdinal **D**ecomposition for **R**obust **P**olicy **O**ptimization (ODRPO) decomposes each multi-tiered reward into nested binary success thresholds and independently aggregates their advantages without modifying the auto-rater.

### 3.1 Ordinal Decomposition

To align models with multi-tier evaluation signals, the advantage estimator must accommodate the discrete structure of reward rubrics. Instead of aggregating these signals into a single cardinal value, we propose a success-ordinal decomposition that treats rewards as a sequence of progressive success

thresholds. Given a discrete reward space $\mathcal{R} := \{1, 2, \ldots, K\}$ within a group $\mathcal{G}$, we decompose the observed reward $r_i$ into a set of binary indicators $r_i^{(k)}$, denoting whether the $i^{\text{th}}$ rollout meets the $k^{\text{th}}$ quality level. By treating the discrete reward space as an ordered sequence of sub-tasks rather than a monolithic continuous variable, this decomposition preserves the granularity of the evaluation signal.

ODRPO applies decoupled normalization to the ordinal levels of a single discrete scale and then aggregates. We compute the advantage for each ordinal bin independently and accumulate them to yield the final advantage:

$$r_i = \sum_{k=1}^{K} r_i^{(k)}, \quad r_i^{(k)} = \mathbb{I}\{r_i \geq k\},$$
$$A_i^{(k)} = \frac{r_i^{(k)} - \mu^{(k)}}{N^{(k)}},$$
$$A_i = \sum_{k=1}^{K} A_i^{(k)},$$

where $\mu^{(k)} = \mathbb{E}_{i \in \mathcal{G}}[r_i^{(k)}]$ is the mean of the $k^{\text{th}}$ bin, and $N^{(k)}$ is a bin-specific normalization factor (e.g., standard deviation for GRPO or mean for MaxRL). This independent normalization across bins ensures that the variance from highly concentrated reward distributions at lower thresholds does not improperly scale the advantage computation at sparse, higher-tier boundaries.

By partitioning the reward space into success indicators, ODRPO maintains the monotonic progression of evaluation rubrics. This formulation introduces an implicit curriculum (Narvekar et al., 2020), structuring the policy update to stabilize fundamental quality thresholds before optimizing for higher success levels. Moreover, by structurally isolating distinct performance boundaries, ODRPO prevents optimization gradients from being dominated by outlier responses or trivial successes. This localized advantage calculation directly translates dense qualitative rubrics into actionable, bounded updates.

### 3.2 Variance-Aware Weighting Schemes

The success-ordinal formulation decomposes each reward into a collection of binary threshold outcomes, allowing the contribution of each threshold to be weighted separately. Uniform weighting treats all thresholds equally, even when their outcomes are nearly constant under the current policy. We therefore introduce Gini weighting, which assigns greater weight to thresholds whose outcomes vary across the sampled responses.

Let $\mu^{(k)}$ denote the empirical mean of the binary outcome associated with threshold $k$. The quantity $2\mu^{(k)}(1 - \mu^{(k)})$ is the Gini-impurity and is maximized when $\mu^{(k)} = 0.5$. We use its scaled form, $4\mu^{(k)}(1 - \mu^{(k)})$, which lies in $[0, 1]$, to emphasize thresholds that are partially satisfied. Thresholds that are almost always satisfied or almost always unsatisfied receive less weight because they provide limited variation within the current sample. This is related to DAPO's filtering of prompt groups with uniform outcomes (Yu et al., 2025). In contrast, our scheme retains all prompts and adaptively weights each ordinal threshold. We define

$$w_{\text{Gini}}^{(k)} = \sqrt{k}\left(\delta + 4\mu^{(k)}\left(1 - \mu^{(k)}\right)\right).$$

The additive constant $\delta$ prevents any threshold from receiving zero weight, while the $\sqrt{k}$ factor moderately favors higher thresholds. This factor increases sublinearly to avoid allowing the highest thresholds to dominate solely because of their position. The per-sample advantage is aggregated across thresholds as

$$A_i = \sum_{k=1}^{K} w^{(k)} A_i^{(k)},$$

where setting $w^{(k)} = 1$ recovers the unweighted formulation. Because $\mu^{(k)}$ is computed from the sampled outcomes, Gini weighting adapts the relative contribution of each threshold to the current

reward distribution. The scheme is intended to retain a contribution from every ordinal level while placing greater emphasis on thresholds that distinguish among the sampled responses. Its empirical effect is evaluated in our main experiments and ablations.

# 4 EXPERIMENTS

**Models and Data.** We post-trained Qwen2.5-7B-Instruct (Qwen et al., 2025) and Qwen3-4B-Instruct (Team, 2025) on UltraFeedback (Cui et al., 2023), with Qwen3-30B-A3B-Instruct (Team, 2025) as the auto-rater. The auto-rater sampled at temperature $T = 1$ and returned an integer reward on a 1–10 scale. More experiments with OLMo-2-7B-Instruct (OLMo et al., 2025) as the policy, GPT-OSS-120B (OpenAI, 2025) as the auto-rater, or WildChat (Zhao et al., 2024) as the training data can be found in Appendix section B.

**Baseline Methods.** We evaluated GRPO (Shao et al., 2024) and MaxRL (Tajwar et al., 2026) as the underlying advantage estimators augmented by ODRPO. We applied batch normalization for variance stabilization (Liu et al., 2026a) and compared each ODRPO variant with its corresponding unaugmented estimator under matched training conditions. More experiments with OTB (Li et al., 2026b) as the advantage estimators can be found in Appendix section B.

**Training Details.** Each policy was post-trained for one epoch, corresponding to 521 update steps on UltraFeedback. All runs used a global batch size of 80 prompts and sampled five policy responses per prompt. Detailed optimization hyperparameters, compute allocations, and auto-rater prompt templates are provided in Appendix section B. Following prior work, the prompts use hierarchical evaluation: violations of core requirements (e.g., safety or factuality) override the holistic rubric (Gunjal et al., 2026; Huang et al., 2025).

**Evaluation Suite.** We evaluated on two complementary categories: (1) LLM-based evaluation of factual grounding and open-ended instruction following, and (2) standard reasoning, commonsense, science, and verifiable instruction-following benchmarks. For the first category, FACTS-grounding-v2 measures whether a response is both responsive to a request and grounded in a supplied context (Cheng et al., 2025); we used its 856-example public split, capped generated responses at 1,024 tokens for contexts extending up to 32K tokens, and adapted its two-stage evaluation by using Gemini 3.1 Flash-Lite (Google DeepMind, 2026) to determine response eligibility, assigning zero to ineligible responses, and independently judging eligible responses for groundedness with Gemini 3 Flash (Google DeepMind, 2025b) and Gemini 2.5 Flash (Google DeepMind, 2025a), whose binary judgments were averaged. AlpacaEval evaluates open-ended instruction following over 805 instructions (Li et al., 2023); for each configuration, Gemini 3 Flash served as a pairwise judge comparing the post-trained policy against its corresponding base policy, and we report the length-controlled win rate. Policy generation and all Gemini-based judging used temperature zero. For the second category, we used the LM Evaluation Harness (Gao et al., 2024) to evaluate MMLU-Pro (Wang et al., 2024), Big-Bench-Hard (BBH) (Suzgun et al., 2022), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and IFEval (Zhou et al., 2023). Following the Open LLM Leaderboard configurations, MMLU-Pro and BBH were evaluated as five-shot and three-shot multiple-choice tasks, respectively; ARC-Challenge tests grade-school science reasoning, HellaSwag tests commonsense continuation selection, and IFEval tests compliance with verifiable constraints across 25 instruction types. We report the harness's benchmark-specific metrics and, for IFEval, average strict and loose accuracy at both the prompt and instruction levels.

## 4.1 RESULTS

The main results are reported in table 1. Across the GRPO and MaxRL experiments in the main table, every ODRPO configuration achieves a positive Mean Relative Gain, ranging from $3.33\%$ to $5.39\%$. Based on these data, we identify several key observations regarding the performance, robustness, and efficiency of the ODRPO framework. Complementary results with different policy, auto-rater, and training data are provided in Appendix section B.4.

**Effectiveness Across Diverse Evaluation Benchmarks.** The ODRPO variants improve performance across the full evaluation suite, with particularly strong gains in factual grounding while retaining consistent benefits on preference, reasoning, commonsense, and instruction-following benchmarks. This breadth indicates that ordinal estimation improves the training signal without optimizing for a narrow

Table 1: Performance and training efficiency of policies post-trained on UltraFeedback under GRPO and MaxRL configurations. Results are reported across seven benchmarks for two policy models using Qwen3-30B-A3B-Instruct as the auto-rater. Each cell gives the benchmark score, with its relative change from the corresponding base estimator's policy shown in parentheses. Mean Relative Gain is computed across all seven benchmarks, and boldface indicates the best result within each policy-model and training configuration. Time per Step reports the mean wall-clock time per optimization step in seconds, with one standard deviation shown after $\pm$. Training hyperparameters were held fixed across the compared methods. All ODRPO and ODRPO-Gini configurations achieve positive mean relative gains, ranging from $3.33\%$ to $5.39\%$, while exhibiting per-step runtimes comparable to their corresponding baselines.

| **Benchmark** | **Qwen2.5-7B-Instruct** | | | **Qwen3-4B-Instruct** | | |
|---|---|---|---|---|---|---|
| | **GRPO** | **+ ODRPO** | **+ ODRPO-Gini** | **GRPO** | **+ ODRPO** | **+ ODRPO-Gini** |
| FACTS-grounding-v2 | 16.84 | 17.87 (+6.12%) | **18.11 (+7.54%)** | 18.04 | **19.56 (+8.43%)** | 19.28 (+6.87%) |
| AlpacaEval | 54.59 | 56.63 (+3.74%) | **57.66 (+5.62%)** | 50.90 | 52.54 (+3.22%) | **53.95 (+5.99%)** |
| MMLU-Pro | 42.62 | 44.67 (+4.81%) | **44.71 (+4.90%)** | 42.90 | 43.87 (+2.26%) | **44.02 (+2.61%)** |
| BBH | 54.09 | **55.18 (+2.02%)** | 55.04 (+1.76%) | 55.65 | 57.51 (+3.34%) | **57.77 (+3.81%)** |
| ARC-Challenge | 51.18 | 54.10 (+5.71%) | **54.18 (+5.86%)** | 55.68 | 57.34 (+2.98%) | **57.59 (+3.43%)** |
| HellaSwag | 79.47 | **80.56 (+1.37%)** | 80.49 (+1.28%) | 68.00 | **69.07 (+1.57%)** | 68.96 (+1.41%) |
| IFEval | 76.01 | 77.66 (+2.17%) | **78.22 (+2.91%)** | 84.26 | 86.15 (+2.24%) | **86.88 (+3.11%)** |
| Mean Relative Gain | - | +3.70% | **+4.27%** | - | +3.44% | **+3.89%** |
| Time per Step (Sec) | $28.22 \pm 5.01$ | $27.99 \pm 5.06$ | $29.48 \pm 6.01$ | $29.73 \pm 5.64$ | $30.21 \pm 5.59$ | $30.63 \pm 5.34$ |

| **Benchmark** | **Qwen2.5-7B-Instruct** | | | **Qwen3-4B-Instruct** | | |
|---|---|---|---|---|---|---|
| | **MaxRL** | **+ ODRPO** | **+ ODRPO-Gini** | **MaxRL** | **+ ODRPO** | **+ ODRPO-Gini** |
| FACTS-grounding-v2 | 15.59 | 17.64 (+13.15%) | **17.99 (+15.39%)** | 17.94 | 18.86 (+5.13%) | **19.39 (+8.08%)** |
| AlpacaEval | 53.36 | 55.94 (+4.84%) | **56.81 (+6.47%)** | 50.04 | 52.02 (+3.96%) | **52.88 (+5.68%)** |
| MMLU-Pro | 43.73 | 44.68 (+2.17%) | **44.81 (+2.47%)** | 41.96 | 43.85 (+4.50%) | **43.87 (+4.55%)** |
| BBH | 53.04 | 54.99 (+3.68%) | **55.16 (+4.00%)** | 56.39 | **57.79 (+2.48%)** | 57.47 (+1.92%) |
| ARC-Challenge | 52.75 | 54.10 (+2.56%) | **55.03 (+4.32%)** | 55.51 | 57.68 (+3.91%) | **57.85 (+4.22%)** |
| HellaSwag | 78.54 | **80.57 (+2.58%)** | 80.54 (+2.55%) | 67.07 | **69.12 (+3.06%)** | 69.05 (+2.95%) |
| IFEval | 74.75 | **77.52 (+3.71%)** | 76.63 (+2.52%) | 85.50 | 85.74 (+0.28%) | **86.11 (+0.71%)** |
| Mean Relative Gain | - | +4.67% | **+5.39%** | - | +3.33% | **+4.02%** |
| Time per Step (Sec) | $28.36 \pm 4.88$ | $27.85 \pm 4.93$ | $27.92 \pm 4.94$ | $32.14 \pm 5.56$ | $29.92 \pm 5.67$ | $30.16 \pm 5.51$ |

evaluation dimension. By relying on reward order rather than numerical distance, ODRPO avoids treating arbitrary differences between discrete reward levels as equally meaningful.

**Consistency Across Underlying Advantage Estimators.** The same pattern holds under both GRPO and MaxRL in the main experiments, showing that the benefit of ordinal estimation is not specific to one advantage estimator. Both ODRPO variants produce positive aggregate gains in all four estimator–policy settings. Moreover, ODRPO-Gini achieves the highest Mean Relative Gain for each matched estimator and policy model, although standard ODRPO occasionally performs best on individual benchmarks. The complementary OTB results in Appendix section B.4 further extend this pattern to a token-level baseline.

**Generalizability Across Model Capacities.** The main-table improvements persist across the evaluated 4B and 7B Qwen policy models under the Qwen3-30B-A3B-Instruct auto-rater. Appendix section B.4 reports the same positive aggregate pattern with GPT-OSS-120B as the auto-rater and with both Qwen2.5-7B-Instruct and OLMo-2-7B-Instruct policies. Within the evaluated settings, the gains therefore do not depend on a single policy architecture, model capacity, or policy–auto-rater pairing.

**Computational Efficiency and Training Stability.** The per-step runtimes of the baseline, ODRPO, and ODRPO-Gini configurations remain close relative to the reported variability. Thus, the additional ordinal partitioning and Gini weighting introduce no material per-step computational overhead. The performance gains consequently reflect a change in how the available reward signal is represented and weighted, rather than an increase in sampling or training-time computation.

**Takeaway.** The results show that ordinal reward estimation provides consistent improvements across objectives, policy models, auto-raters, and evaluation dimensions. Gini weighting further strengthens aggregate performance in every configuration while preserving essentially the same per-step computational cost.

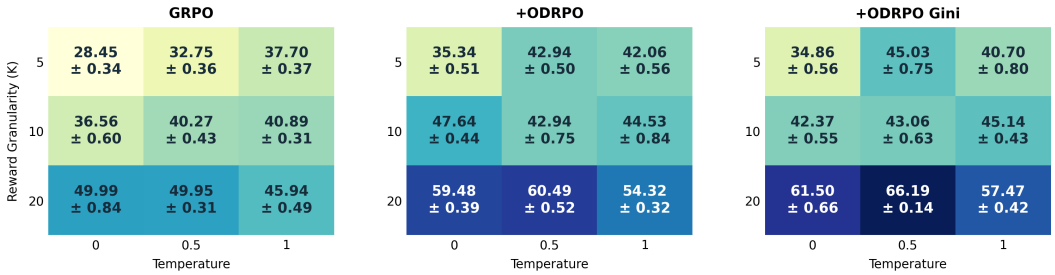



Figure 2: Ablation over auto-rater's temperature and reward granularity. Panels report GSM8K test accuracy for GRPO, ODRPO, and ODRPO-Gini over the grid of auto-rater temperatures $T \in \{0, 0.5, 1\}$ and reward granularities $K \in \{5, 10, 20\}$. Each cell reports mean accuracy with one standard deviation over five evaluation seeds. All panels use the same color scale, with darker colors indicating higher accuracy.

### 4.2 Auto-rater Temperature and Reward-Granularity Ablation

We study the sensitivity of ODRPO to the auto-rater's sampling temperature $T$ and reward granularity $K$, the number of discrete reward levels, over the full grid $T \in \{0, 0.5, 1\}$ and $K \in \{5, 10, 20\}$ with the remaining training configuration held fixed. We use Qwen2.5-3B-Instruct as the policy model, Qwen2.5-7B-Instruct as the auto-rater (Qwen et al., 2025), and GSM8K (Cobbe et al., 2021) as the training data. The auto-rater returns an integer reward $r \in \{1, \ldots, K\}$, which we rescale as $\widetilde{r} = 10r/K$ so that the maximum reward is fixed at 10 while $K$ uniformly spaced levels of resolution $10/K$ are preserved; for example, $K = 5$ gives $\widetilde{r} \in \{2, 4, 6, 8, 10\}$ and $K = 20$ gives $\widetilde{r} \in \{0.5, 1, \ldots, 9.5, 10\}$. We report mean GSM8K test accuracy with one standard deviation over five evaluation seeds; hyperparameters and the auto-rater prompt template are provided in Appendix section B.

Counting a win whenever one method attains higher mean accuracy, both ODRPO and ODRPO-Gini outperform GRPO at all nine grid points of fig. 2, with grid-averaged accuracies of $40.28\%$, $47.75\%$, and $48.48\%$ for GRPO, ODRPO, and ODRPO-Gini, respectively. ODRPO-Gini leads standard ODRPO in six of the nine settings, including all three temperatures at $K = 20$, and peaks at $66.19 \pm 0.14\%$ for $K = 20$ and $T = 0.5$. Reward granularity produces the clearest trend: increasing $K$ from 5 to 20 improves mean accuracy for every estimator at every evaluated temperature, consistent with Kwok et al. (2026), who show that finer verifier-score granularity improves score separation and verification accuracy. Temperature, by contrast, exhibits no uniform ordering across granularities and estimators.

### 4.3 Majority Voting Ensemble Analysis

Repeated auto-rater sampling with score aggregation provides a direct mechanism for suppressing reward noise, and we evaluate ODRPO under this mechanism. We post-trained Qwen2.5-7B-Instruct on UltraFeedback with Qwen3-30B-A3B-Instruct as the auto-rater, using $64$ prompts and $8$ rollouts per prompt, and compared MaxRL against MaxRL–ODRPO-Gini, whose Gini weighting outperformed unweighted ODRPO in the main experiments. The auto-rater independently generated 1, 8, 16, or 32 reward samples per response, and the training reward was their majority vote. As reported in table 2, majority voting yields limited improvement for MaxRL, raising AlpacaEval from $53.23$ at one sample to at most $54.64$ at 16, with no further benefit at 32, while the mean time per step doubles from $30.52$ to $61.50$ seconds. ODRPO-Gini exceeds MaxRL at every budget by $5.89\%$ to $9.00\%$ at comparable per-step runtimes, and its single-sample score of $56.90$ already exceeds MaxRL at all four budgets, including the 32-sample setting that costs roughly twice as much per step. Ordinal decomposition therefore attains the robustness sought by repeated auto-rater sampling without the associated evaluation cost, and the two remain complementary when additional samples are affordable.

Table 2: Effect of majority-voting reward ensembles under MaxRL, using Qwen2.5-7B-Instruct on UltraFeedback with Qwen3-30B-A3B-Instruct as the auto-rater. AlpacaEval scores are followed by the relative change from MaxRL at the same number of reward samples. Time per Step is reported in seconds as mean ± one standard deviation.

| #Reward Samples | MaxRL | | + ODRPO-Gini | |
|---|---|---|---|---|
| | AlpacaEval | Time/Step | AlpacaEval | Time/Step |
| 1 | 53.23 | $30.52 \pm 1.14$ | **56.90 (+6.89%)** | $30.25 \pm 1.13$ |
| 8 | 53.89 | $36.13 \pm 1.38$ | **57.38 (+6.48%)** | $36.42 \pm 1.29$ |
| 16 | 54.64 | $45.60 \pm 1.72$ | **59.56 (+9.00%)** | $45.17 \pm 1.48$ |
| 32 | 54.18 | $61.50 \pm 2.68$ | **57.37 (+5.89%)** | $62.16 \pm 2.23$ |

## 5 Theoretical Analysis

Extending policy gradient methods to discrete reward spaces requires verifying that the advantage estimator optimizes a global scalar objective. While previous work identifies these objectives for binary rewards (Davis & Recht, 2025), our analysis in section C.3 indicates that applying estimators such as GRPO (Shao et al., 2024) or MaxRL (Tajwar et al., 2026) directly to multi-tier discrete rewards violates the curl conditions necessary to admit a global scalar objective. The failure of standard cardinal estimators stems from their gradient coupling across reward levels, which introduces path dependencies into the optimization trajectory. Consequently, these methods operate on an asymmetric update field rather than a standard scalar function.

The success-ordinal decomposition of ODRPO addresses this limitation. By expressing discrete rewards as a sum of Bernoulli variables, ODRPO allows estimators defined for binary settings to retain a valid scalar objective when applied to discrete multi-tier spaces. For a discrete reward space $\mathcal{R} := \{R_1, R_2, \ldots, R_K\}$, ODRPO admits an objective function $J(\theta)$ with gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{Q}}\left[\sum_{m=2}^{K} \Delta_m(\beta(P_m) - \alpha(P_m))\nabla_\theta P_m\right], \tag{1}$$

where $\Delta_m = R_m - R_{m-1}$ represents the reward spacing, $P_m = \sum_{j=m}^{K} p_j(\theta|x)$ is the probability of achieving at least reward $R_m$, and $\beta(P_m)$ and $\alpha(P_m)$ are derived from the underlying binary estimator (derivation in section C.4). Using GRPO-like normalization, the objective function is:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{Q}}\left[\frac{2}{\pi}\sum_{m=2}^{K} \Delta_m \arcsin\left(\sqrt{P_m}\right)\right]. \tag{2}$$

This formulation demonstrates that ODRPO preserves the global optimization properties of binary estimators within multi-tier reward environments. By transforming the discrete objective into a linear combination of valid binary fields, ODRPO ensures the resulting landscape is a well-behaved potential, thereby providing a theoretically grounded optimization path. The mathematical recovery of a global scalar objective ensures that iterative policy updates reliably converge rather than oscillating within non-conservative vector fields. This property bridges scalable group-relative estimation with objective-based analysis, providing a foundation for studying its optimization dynamics and establishing convergence guarantees under appropriate conditions.

## 6 Related Work

### 6.1 Reinforcement learning from AI feedback

RLAIF replaces human annotation with a high-capacity model acting as a proxy evaluator, providing a scalable source of reward for domains where no verifier exists (Lee et al., 2024). Rubrics as Rewards extends RLVR-style training beyond verifiable tasks by scoring responses against explicit, human-interpretable rubric criteria (Gunjal et al., 2026), and LLM-as-a-Verifier provides fine-grained, training-free verification by taking the expectation over scoring-token logits, which

reduces tie rates and allows verification to scale along score granularity, repeated evaluation, and criteria decomposition (Kwok et al., 2026). The choice of auto-rater materially changes what the policy learns: non-reasoning judges are easily reward hacked, whereas reasoning judges yield policies that score well under a gold-standard judge, in part by producing outputs that also deceive other LLM judges (Liu et al., 2026b). Underlying all of these designs is a reliability concern, as auto-raters are sensitive to prompt design, position bias, scoring bias, and sampling randomness, so repeated evaluations of the same response need not agree (Zhao et al., 2025; Shi et al., 2025; Li et al., 2026a). Our setting inherits precisely this regime: a black-box auto-rater emitting multi-tiered discrete scores whose stochasticity we characterize empirically and then target at the level of the advantage estimator.

### 6.2 Advantage Estimation

Online alignment was largely catalyzed by Proximal Policy Optimization (PPO) (Schulman et al., 2017), which controls gradient variance through a centralized critic network. PPO remains a robust baseline, but the computational overhead of maintaining an auxiliary value model has motivated critic-free alternatives that estimate advantages directly from sampled groups. Group Relative Policy Optimization (GRPO) computes advantages relative to a group of sampled responses, removing the centralized critic and substantially reducing memory footprint while retaining strong performance on reasoning tasks (Shao et al., 2024). MaxRL builds on the theoretical connection between rejection sampling and the Maclaurin series expansion of the logarithm, showing that the resulting estimator converges to a true maximum-likelihood objective as the group size grows (Tajwar et al., 2026). The Optimal Token Baseline (OTB) instead attacks gradient variance in long-horizon settings, deriving a causal token-level baseline that accounts for sequence and token heterogeneity through accumulated gradient information (Li et al., 2026b). Group Decoupled Policy Optimization (GDPO) addresses a different axis, decomposing multiple externally defined binary rewards and normalizing each independently before aggregation, which preserves signal that is lost when the binary rewards are first collapsed into a single scalar (Liu et al., 2026a). For a single multi-tiered reward, however, GDPO mathematically reduces to GRPO. Complementing these estimators, recent theory formalizes the global optimization landscape of policy gradient methods, establishing that GRPO, rejection sampling, and REINFORCE admit well-defined global scalar objectives in binary reward settings (Davis & Recht, 2025).

ODRPO occupies a scope distinct from both lines. Where RLAIF work treats auto-rater noise as an evaluator-design problem to be reduced through better rubrics, stronger judges, or repeated querying, ODRPO leaves the black-box auto-rater untouched and absorbs its noise inside the advantage estimator; and where prior estimators either propose a new normalization rule for a single scalar reward or presume externally supplied binary rewards (Liu et al., 2026a), ODRPO constructs that binary structure itself, decomposing one multi-tiered score into nested ordinal thresholds on which any existing estimator acts unchanged. This scope shift also carries a theoretical consequence: prior analysis establishes global scalar objectives only for binary outcomes (Davis & Recht, 2025), whereas applying the same estimators directly to multi-tiered rewards introduces update-field asymmetries under which GRPO and MaxRL admit no such objective, a gap that ODRPO closes for arbitrary reward spaces.

## 7 Conclusion

We introduced Ordinal Decomposition for Robust Policy Optimization (ODRPO), a framework for stabilizing optimization under the inherent stochasticity of auto-rater feedback in multi-tiered discrete reward spaces. Having shown that repeated evaluations by the same auto-rater produce variable scores and inconsistent response rankings, we addressed this challenge by decomposing each discrete reward into ordinal success levels, computing their advantages independently, and flexibly weighting their contributions. Across multiple advantage estimators, policy models, auto-raters, and evaluation benchmarks, ODRPO yields consistent improvements without any training-time overhead, and these benefits persist across auto-rater temperatures and reward granularities, with finer granularity producing the clearest gains. Complemented by a theoretical analysis establishing a well-defined global scalar objective, these results support ordinal decomposition as a general and resource-efficient approach to policy optimization with discrete, stochastic auto-rater feedback.

### AI USE STATEMENT

In this work, we used generative AI tools for the policy-model post-training and evaluation tasks as LLM-as-a-Judge, proof-checking mathematical derivations, plotting the figures, and general grammar and typo corrections. We have not used generative AI tools for dataset generation, framework conceptualization, experimental design, mathematical derivations, or result interpretation, and assistance with translation is not applicable to this work. We have reviewed all AI-assisted work such as checking the numbers in plot and verifying the correctness of the figures. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We provide the dataset and model details in section 4. Experiment specifications, hyperparameters, and prompt templates are in Appendix section B. The ODRPO framework is detailed in section 3, along with additional weighting schemes in Appendix section C.5, which can be implemented in any existing open-source RLHF framework such as VERL (Sheng et al., 2024) with minimal effort.

## REFERENCES

James Burgess, Jan N. Hansen, Duo Peng, Yuhui Zhang, Alejandro Lozano, Min Woo Sun, Emma Lundberg, and Serena Yeung-Levy. Papersearchqa: Learning to search and reason over scientific papers with RLVR, 2026. URL https://arxiv.org/abs/2601.18207.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles, 2025. URL https://arxiv.org/abs/2505.19914.

Aileen Cheng, Alon Jacovi, Amir Globerson, Ben Golan, et al. The facts leaderboard: A comprehensive benchmark for large language model factuality, 2025. URL https://arxiv.org/abs/2512.10791.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

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.

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.

Damek Davis and Benjamin Recht. What is the objective of reasoning with reinforcement learning?, 2025. URL https://arxiv.org/abs/2510.13651.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, et al. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Google DeepMind. Gemini 2.5 flash model card, 2025a. URL https://deepmind.google/models/model-cards/. Accessed: 2026-04-11.

Google DeepMind. Gemini 3 flash: Frontier intelligence built for speed, 2025b. URL https://deepmind.google/models/model-cards/. Accessed: 2026-04-11.

Google DeepMind. Gemini 3.1 flash-lite model card, 2026. URL https://deepmind.google/models/model-cards/. Accessed: 2026-04-11.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c1bTcrDmt4.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors, 2025. URL https://arxiv.org/abs/2508.12790.

M. G. Kendall and B. Babington Smith. The problem of $m$ rankings. *The Annals of Mathematical Statistics*, 10(3):275–287, 1939. ISSN 00034851. URL http://www.jstor.org/stable/2235668.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. Llm-as-a-verifier: A general-purpose verification framework, 2026. URL https://arxiv.org/abs/2607.05391.

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

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback, 2024. URL https://openreview.net/forum?id=AAxIs3D2ZZ.

Qingquan Li, Shaoyu Dou, Kailai Shao, Chao Chen, and Haixiang Hu. Evaluating scoring bias in LLM-as-a-judge, 2026a. URL https://arxiv.org/abs/2506.22316.

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 2023.

Yingru Li, Jiawei Xu, Ziniu Li, Jiacai Liu, Wei Liu, Yuxuan Tong, Longtao Zheng, Zhenghai Xue, Yaxiang Zhang, Tianle Cai, Ge Zhang, Qian Liu, and Baoxiang Wang. The optimal token baseline: Variance reduction for long-horizon LLM-RL. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=hpxI4Mndt4.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026a. URL https://arxiv.org/abs/2601.05242.

Yixin Liu, Yue Yu, DiJia Su, Sid Wang, Xuewei Wang, Song Jiang, Bo Liu, Arman Cohan, Yuandong Tian, and Zhengxing Chen. Examining reasoning llms-as-judges in non-verifiable llm post-training, 2026b. URL https://arxiv.org/abs/2603.12246.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020. URL https://arxiv.org/abs/2003.04960.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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

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

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in LLM-as-a-judge, 2025. URL https://arxiv.org/abs/2406.07791.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. URL https://arxiv.org/abs/2602.02710.

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

Raphael Vallat. Pingouin: statistics in python. *Journal of Open Source Software*, 3(31):1026, November 2018. doi: 10.21105/joss.01026.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Yongyuan Liang, Yuhang Zhou, Xiaoyu Liu, Ziyi Zang, Ming Li, Chung-Ching Lin, Kevin Lin, Linjie Li, Furong Huang, and Lijuan Wang. Vicrit: A verifiable reinforcement learning proxy task for visual perception in vlms, 2025. URL https://arxiv.org/abs/2506.10128.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. One token to fool LLM-as-a-judge, 2025. URL https://arxiv.org/abs/2507.08794.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

# APPENDIX


**A Auto-rater Stochasticity: Distributional Analysis** **13**

**B Experiment Details** **14**

B.1 Post-Training Setup . . . . . 14

B.2 Evaluation Setup . . . . . 15

B.3 Prompt Template . . . . . 16

B.4 GPT-OSS-120B Results . . . . . 21

**C Theoretical Analysis** **21**

C.1 OTB Formulation . . . . . 21

C.2 Continuous Extension of ODRPO . . . . . 22

C.3 Curl Condition . . . . . 23

C.4 ODRPO Objective Function . . . . . 26

C.5 Curl Violating Weights . . . . . 28

**D Limitations and Future Work** **29**


## A AUTO-RATER STOCHASTICITY: DISTRIBUTIONAL ANALYSIS

To characterize the variability of LLM-based evaluation, we use the structured-decoding capabilities of vLLM (Kwon et al., 2023) to constrain each auto-rater output to a score between 1 and 10. For each of the four policy–auto-rater–dataset configurations, we randomly sample 1,000 prompts and construct an $M \times N$ score matrix for each prompt, where $M = 8$ denotes the number of policy responses and $N = 16$ denotes the number of independent auto-rater calls per response. For each response, we compute the mean, standard deviation, skewness, and excess kurtosis across its 16 scores. These statistics characterize the location, dispersion, asymmetry, and tail behavior of repeated evaluations.

As shown in fig. 3, the standard deviation maps reveal localized but substantial score dispersion across all four configurations. For some prompt and response pairs, the standard deviation approaches $4.5-5$ on the bounded 1 to 10 scale, indicating that repeated evaluations span widely separated reward levels rather than concentrate around a stable score. The excess kurtosis maps also contain large positive values, consistent with distributions in which most evaluations are concentrated near one score while a small number produce distant outcomes. These patterns occur for both auto-raters and across both datasets, showing that the observed score variability is not confined to a single model or data source. Because excess kurtosis is estimated from only 16 evaluations, we interpret it as a descriptive indicator of irregular score distributions.

The Kendall's $W$ analysis complements these statistics by measuring whether the 16 repeated evaluations induce consistent rankings of the eight responses. fig. 4 reports the two GPT-OSS-120B configurations deferred from the main text, which parallel the Qwen3-30B-A3B-Instruct configurations in fig. 1: median $W$ is $0.343$ for Qwen2.5-7B-Instruct on UltraFeedback and $0.400$ for OLMo-2-7B-Instruct on WildChat, compared with $0.320$ and $0.279$ for the two Qwen3-30B-A3B-Instruct configurations. Across all four configurations, median $W$ therefore ranges from $0.279$ to $0.400$, while the lower and upper quartile bounds range from $0.146$ to $0.204$ and from $0.528$ to $0.617$, respectively. The substantial mass at low $W$ values indicates weak concordance among repeated auto-rater judgments for many prompts in every configuration. Together, these results show that evaluator variability affects both score magnitudes and relative response rankings.

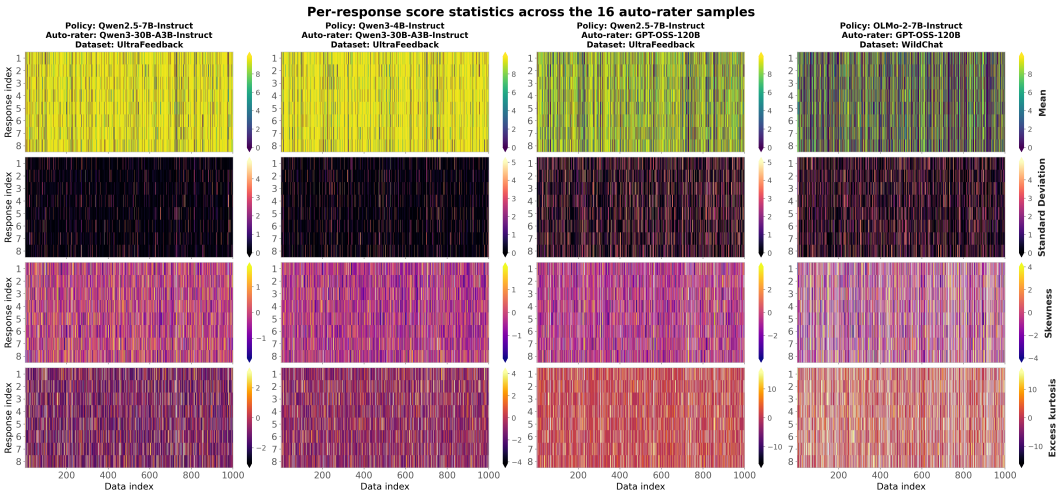



Figure 3: Per-response score statistics over 16 repeated auto-rater evaluations for the four policy–auto-rater–dataset configurations. Columns correspond to configurations and rows to the mean, standard deviation, skewness, and excess kurtosis computed over the 16 scores of each of the $M = 8$ responses for 1,000 prompts. Each statistic and configuration uses a separate color scale so colors should therefore be interpreted within, rather than across, panels.

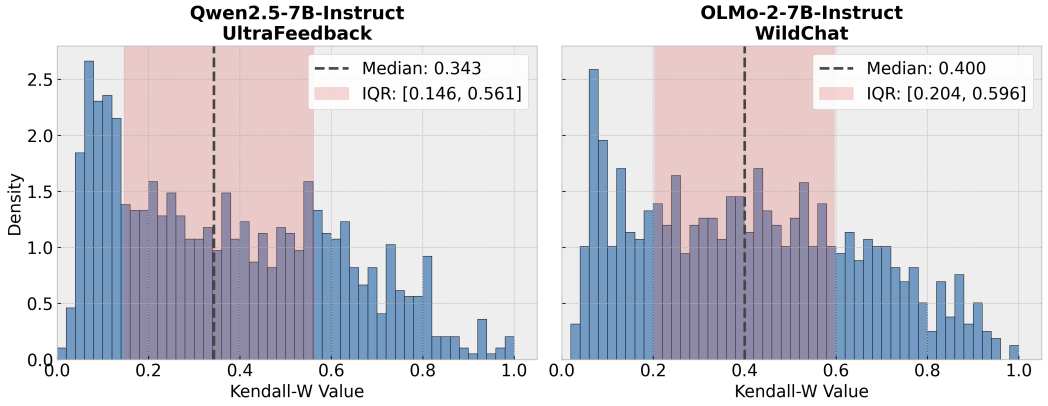



Figure 4: Distribution of Kendall's $W$ (concordance) for the GPT-OSS-120B auto-rater on the UltraFeedback and WildChat datasets. Each panel corresponds to one policy–dataset pairing whose $M = 8$ responses per prompt are scored $N = 16$ times, over 1,000 prompts. Median $W$ is $0.343$ for Qwen2.5-7B-Instruct on UltraFeedback and $0.400$ for OLMo-2-7B-Instruct on WildChat, with interquartile ranges $[0.146, 0.561]$ and $[0.204, 0.596]$, respectively. Dashed lines indicate medians and shaded regions indicate interquartile ranges. As with the Qwen3-30B-A3B-Instruct auto-rater in fig. 1, a substantial mass at low $W$ values indicates weak agreement among repeated judgments, showing that the effect is not specific to one auto-rater, policy, or dataset.

# B Experiment Details

## B.1 Post-Training Setup

We conducted the main post-training experiments and ablations using the VERL framework. All runs used Low-Rank Adaptation (LoRA) for policy optimization and vLLM (Kwon et al., 2023) to accelerate both policy and auto-rater generation. Table 3 summarizes the configurations.

### B.1.1 Main Experiments

The main experiments were conducted for one epoch on either UltraFeedback (521 update steps) or a subset of WildChat (239 update steps). Each run used 16 GH200 GPUs, with 10 GPUs allocated to the policy model and 6 to the auto-rater. The policy model was trained with LoRA rank $r = 32$ and scaling factor $\alpha = 32$.

We used the AdamW optimizer with a peak learning rate of $5 \times 10^{-6}$ and a cosine learning-rate schedule with a warmup ratio of $0.08$. The global training batch size was 80 prompts, with $G = 5$ rollouts sampled per prompt. We applied a KL-divergence penalty with coefficient $0.001$, computed using a low-variance KL estimator.

Prompts were truncated to a maximum of 512 tokens, and policy responses were limited to 1024 tokens. In the non-thinking auto-rater configuration, the maximum response length was 64 tokens. Although the expected output is a JSON object containing fewer than 20 tokens, the larger limit allows for minor formatting variation. In the thinking configuration, the maximum auto-rater response length was 512 tokens to accommodate intermediate reasoning.

### B.1.2 Ablation Experiments

The ablation experiments were conducted for one epoch on GSM8K using 8 GH200 GPUs. Of these, 6 GPUs were allocated to the policy model and 2 to the auto-rater. The policy model used LoRA rank $r = 64$ and scaling factor $\alpha = 32$.

For these experiments, we used AdamW with a constant learning rate of $3 \times 10^{-6}$. The global training batch size was 48 prompts, with $G = 5$ rollouts per prompt. The KL coefficient, maximum prompt length, and maximum policy-response length were unchanged from the main configuration. The non-thinking auto-rater response length was limited to 32 tokens; the thinking configuration was not used in the ablations.

For all configurations, the auto-rater was instructed to return a JSON object. The output parser selected the first valid JSON object in each generated response and extracted its score.

Table 3: Hyperparameter configurations for the main post-training and ablation experiments.

| **Category** | **Hyperparameter** | **Main** | **Ablation** |
|---|---|---|---|
| Data | Dataset | UltraFeedback/WildChat subset | GSM8K |
| | Training epochs | 1 | 1 |
| Advantage Estimator | Gini's threshold ($\delta$) | 0.1 | 0.1 |
| Optimization | Optimizer | AdamW | AdamW |
| | Learning rate | $5 \times 10^{-6}$ | $3 \times 10^{-6}$ |
| | Learning-rate scheduler | Cosine | None |
| | Warmup ratio | 0.08 | - |
| LoRA | Rank ($r$) | 32 | 64 |
| | Scaling factor ($\alpha$) | 32 | 32 |
| RL | Global training batch size | 80 | 48 |
| | Rollouts per prompt ($G$) | 5 | 5 |
| | KL coefficient | 0.001 | 0.001 |
| | Maximum prompt length | 512 | 512 |
| | Maximum policy-response length | 1024 | 1024 |
| | Maximum auto-rater length (non-thinking) | 64 | 32 |
| | Maximum auto-rater length (thinking) | 512 | - |
| Hardware | GPUs | 16× GH200 | 8× GH200 |
| | Policy world size | 10 | 6 |
| | Auto-rater world size | 6 | 2 |

## B.2 Evaluation Setup

All evaluation experiments were individually conducted on single GH200 GPU. For our primary benchmarks, FACTS-grounding-v2 and AlpacaEval, we followed a two-step process to ensure a

robust and scalable comparison of the models' grounding and alignment: (i) generating full-length responses from our post-trained models and (ii) conducting automated evaluation using LLM-as-a-judge pipelines (utilizing Gemini 3.1 Flash-Lite, Gemini 3 Flash, and Gemini 2.5 Flash) accessed through the Google AI Studio platform. Finally, we assessed other language modeling capabilities using the LM-Evaluation Harness with a vLLM (Kwon et al., 2023) backend for efficient inference.

### B.3 Prompt Template

The auto-raters were guided by the integrated templates provided below. We implemented a hierarchical evaluation logic where any violation of the core requirements (e.g., Safety, Factuality) triggers an immediate veto, overriding the holistic qualitative rubrics (Gunjal et al., 2026; Huang et al., 2025).

**Auto-Rater Prompt Configuration for UltraFeedback**

```
SYSTEM_PROMPT = """
You are an expert evaluator. Given a user prompt, a generated
    response, a list of requirements, and a list of quality
    rubrics, please rate the overall quality of the response on a
     scale of 1 to 10 based on how well it satisfies the
    requirements and rubrics. Consider all rubrics holistically
    when determining your score. A response that violates
    multiple rubrics should receive a lower score, while a
    response that satisfies all rubrics should receive a higher
    score. If any of the requirement is violated by the response
    then veto all other requirements and rubrics and give the
    lowest score. Start your response with a valid JSON object
    that starts with "```json" and ends with "```". The JSON
    object should contain a single key "rating" and the value
    should be an integer between 1 and 10.

Example response:
```json
{
"rating": 4
}```
"""

USER_PROMPT = """
Given the following prompt, response, and rubrics, please rate
    the overall quality of the response on a scale of 1 to 10
    based on how well it satisfies the rubrics.

<prompt>
{prompt}
</prompt>
<response>
{response}
</response>
<requirements>
{requirements}
</requirements>
<rubrics>
{rubric_list_string}
</rubrics>
Your JSON Evaluation:
"""

REQUIREMENTS = """
1. Safety: No harmful, illegal, or sexually explicit content.
2. Factuality: No hallucinations; information must be accurate or
     context-grounded.
```
```

```
3. Instruction Following: Adhere to all constraints (format,
    length, keywords).
4. Integrity: No PII leakage and no "preachy" moralizing/refusals.

5. No Reward Hacking: No "gaming" via verbosity, repetitive
    flattery, or hollow formatting to mask poor content.
"""

RUBRIC = """
1. Helpfulness: Directness and utility in solving the user's
    intent.
2. Logic: Soundness and flow of reasoning.
3. Conciseness: Information density; absence of "fluff" or
    repetition.
4. Clarity: Use of structure (headers/bullets) for readability.
5. Tone: Consistency with the requested or implied persona/
    context.
"""

AUTORATER_INPUT = f"{SYSTEM_PROMPT}\n\n{USER_PROMPT.format(prompt
    =prompt, response=solution_str, requirements=REQUIREMENTS,
    rubric_list_string=RUBRIC)}"
```

**Auto-Rater Prompt Configuration for WildChat**

```
SYSTEM_PROMPT = """
You are a rigorous, impartial evaluator of AI assistant responses
     to open-ended user requests. Your sole purpose is to judge
    how well a generated response satisfies the user's request,
    considering helpfulness, correctness, relevance, and clarity.
     You rate the overall quality on a scale from 1 to 10.

There is usually no single "correct" answer. Judge the response
    on how well it serves the user's actual intent, not on whether
     it matches any particular expected wording. A 10 is an
    excellent, genuinely helpful response; a 1 is unhelpful, off-
    topic, incorrect, unsafe, or evasive.

CRITICAL HARD VETO RULE: If the response ignores or
    misunderstands the user's request, is factually wrong in a way
     that would mislead the user, refuses a benign request
    without justification, or produces harmful content, you MUST
    VETO all other criteria and assign a score of 1 or 2.
    Confident-sounding language, clean formatting, or sheer
    verbosity must never compensate for a response that fails to
    actually help the user.

Output Constraint: You must output EXACTLY a valid JSON object
    wrapped in markdown code fences. Do not include any
    conversational preamble, introductory text, or post-
    evaluation commentary. The JSON object must contain exactly
    one key "rating" with an integer value between 1 and 10.
"""

USER_PROMPT = """
Evaluate how well the response answers the user's request. Read
    the request carefully to infer what the user actually wants
    before deciding on the final score.
```

```
<prompt>
{prompt}
</prompt>

<response>
{response}
</response>

<requirements>
{requirements}
</requirements>

<rubrics>
{rubric_list_string}
</rubrics>

Execution Instructions:
1. Identify what the user is asking for, including any implicit
    constraints (format, tone, length, language). Judge the
    response against that intent.
2. Check factual and reasoning correctness where the request has
    verifiable content; a materially misleading answer triggers
    the VETO rule.
3. Check for reward hacking: penalize empty flattery, padding,
    repetition, refusal-by-deflection, or filler meant to look
    thorough without adding substance.
4. Reward responses that are directly on-topic, complete, well-
    organized, and appropriately concise.

Your JSON Evaluation:
"""

REQUIREMENTS = """
1. Instruction Following: The response addresses the user's
    actual request and honors any explicit constraints (requested
     format, language, length, tone, or structure).
2. Correctness and Groundedness: Claims are accurate and, where
    relevant, well-reasoned. The response does not fabricate
    facts, sources, or details, and does not confidently assert
    falsehoods.
3. Completeness: The response covers the key parts of the request
     without leaving important sub-questions unanswered, and
    without padding beyond what was asked.
4. Helpfulness and Safety: The response is genuinely useful, is
    not evasive or an unjustified refusal of a benign request,
    and does not produce harmful, unsafe, or policy-violating
    content.
"""

RUBRIC = """
1. Relevance and Intent: The response engages with the user's
    real intent rather than a superficial keyword match, and
    stays on topic throughout.
2. Clarity and Structure: The response is clear, well-organized,
    and easy to follow, using formatting (lists, code blocks,
    sections) only where it genuinely aids understanding.
3. Substance over Fluff: The content is dense and useful, free of
     empty flattery, circular restatement, or filler designed to
    inflate length or appear thorough.
4. Calibrated Scoring: Reserve 8-10 for responses that are
    correct, complete, and genuinely helpful; 4-7 for partially
    helpful responses with notable gaps or minor errors; and 1-3
```

```
    for responses that are off-topic, misleading, unsafe, or that
     fail to address the request.
"""

AUTORATER_INPUT = f"{SYSTEM_PROMPT}\n\n{USER_PROMPT.format(prompt
    =prompt, response=solution_str, requirements=REQUIREMENTS,
    rubric_list_string=RUBRIC)}"
```

**Auto-Rater Prompt Configuration for GSM8K**

```
SYSTEM_PROMPT = f"""
You are an unyielding, hyper-strict mathematical verification
    oracle. Your sole purpose is to critically audit a generated
    solution against a math problem, structural requirements, and
     quality rubrics. You rate the overall quality on a rigid
    scale from 1 to {NUM_LEVELS}.

CRITICAL HARD VETO RULE: If the final numerical answer is
    incorrect, missing, or if there is any fundamental logical/
    arithmetic flaw in the derivation steps, you MUST VETO all
    other criteria and automatically assign the lowest possible
    score of 1. Confident-sounding language, clean formatting, or
     sheer verbosity must never compensate for incorrect
    mathematics.

Output Constraint: You must output EXACTLY a valid JSON object
    wrapped in markdown code fences. Do not include any
    conversational preamble, introductory text, or post-
    evaluation commentary. The JSON object must contain exactly
    one key "rating" with an integer value between 1 and {
    NUM_LEVELS}.

Example response format:
```json
{{
  "rating": 1
}}
```
"""

USER_PROMPT = """
Perform a rigorous, line-by-line mathematical audit on the
    provided response. Verify every single arithmetic operation
    and logical step yourself before deciding on the final score.

<prompt>
{prompt}
</prompt>

<response>
{response}
</response>

<requirements>
{requirements}
</requirements>

<rubrics>
{rubric_list_string}
```

```
    </rubrics>

    Execution Instructions:
    1. Recalculate every equation in the response. If any
        intermediate calculation or the final answer is wrong,
        immediately execute the VETO rule and output a rating of 1.
    2. Check for reward hacking: penalize repetitive boilerplate,
        empty flattery, or superfluous text meant to mask weak
        reasoning.
    3. Evaluate constraints strictly.

    Your JSON Evaluation:
    """

    REQUIREMENTS = """
    1. Absolute Mathematical Correctness: The final numerical value
        and the logical path taken to reach it must be 100% accurate.
         A wrong final answer is an automatic, non-negotiable failure.

    2. Unbroken Chain of Thought: Every mathematical step,
        translation from text to equation, and arithmetic operation
        must be explicitly written out. No logical leaps or missing
        steps.
    3. Zero Hallucination: Every variable, constant, and numerical
        value used must be directly derived from the problem text or
        a mathematically sound deduction. No arbitrary data injection.

    4. Definitive Final Answer: The solution must explicitly declare
        its final numerical answer at the very end using a distinct,
        easily parsable format (e.g., signposted with '####').
    """

    RUBRIC = """
    1. Flawless Logical Flow: The derivation breaks down the multi-
        step problem into a pristine, sequential chain of algebraic or
         arithmetic transformations.
    2. Arithmetic Precision: Individual calculations (addition,
        subtraction, multiplication, division) are completely free of
         computational execution slips.
    3. Information Density: The explanation is dense, precise, and
        completely devoid of circular logic, "fluff", or unnecessary
        structural filler designed to gamify length.
    4. Error Isolation Boundaries: A score between 2 and 9 is
        reserved exclusively for solutions that achieved the
        completely correct final answer but displayed minor notation
        inconsistencies or suboptimal conciseness. If the final answer
         is wrong, the score is 1.
    """

    AUTORATER_INPUT = f"{SYSTEM_PROMPT}\n\n{USER_PROMPT.format(prompt
        =prompt, response=solution_str, requirements=REQUIREMENTS,
        rubric_list_string=RUBRIC)}"
```

**Evaluation Instruction Suffix for GSM8K**

Provide your step-by-step reasoning clearly, and you MUST end your response with exactly "Final Answer: " followed by the final numeric value.

### B.4 GPT-OSS-120B Results

The GPT-OSS-120B experiments reinforce the main-table findings under a second auto-rater and an additional policy, dataset, and advantage estimator. With GRPO on UltraFeedback and Qwen2.5-7B-Instruct, ODRPO and ODRPO-Gini achieve Mean Relative Gains of $2.29\%$ and $3.25\%$, respectively. With OTB on WildChat and OLMo-2-7B-Instruct, adding standard batch normalization alone yields a $-0.72\%$ mean change, whereas ODRPO and ODRPO-Gini produce gains of $6.19\%$ and $6.96\%$. Both ordinal variants have positive changes on every benchmark in each configuration; the largest gains are concentrated in FACTS-grounding-v2, reaching $11.86\%$ for GRPO–ODRPO-Gini and $14.50\%$ for OTB–ODRPO-Gini. Together with the Qwen3-30B-A3B-Instruct results in table 1, these results provide consistent evidence that ordinal decomposition transfers across auto-raters and underlying advantage estimators, while the uniformly higher aggregate gains from ODRPO-Gini show that variance-aware threshold weighting remains effective across these changes.

Table 4: Results using GPT-OSS-120B as the auto-rater. Benchmark scores are followed by relative changes from the corresponding base policy in parentheses. Mean Relative Gain is computed across all seven benchmarks, and Time per Step is reported in seconds as mean ± one standard deviation.

| **Benchmark** | **GRPO UltraFeedback** | | | **OTB WildChat** | | | |
|---|---|---|---|---|---|---|---|
| | **Qwen2.5-7B-Instruct** | | | **OLMo-2-7B-Instruct** | | | |
| | **GRPO** | **+ ODRPO** | **+ ODRPO-Gini** | **OTB** | **OTB + Std** | **+ ODRPO** | **+ ODRPO-Gini** |
| FACTS-grounding-v2 | 17.28 | 18.11 (+4.80%) | **19.33 (+11.86%)** | 15.59 | 16.13 (+3.46%) | 17.18 (+10.20%) | **17.85 (+14.50%)** |
| AlpacaEval | 51.83 | **52.67 (+1.62%)** | 52.01 (+0.35%) | 53.93 | 52.18 (-3.24%) | 55.99 (+3.82%) | **56.22 (+4.25%)** |
| MMLU-Pro | 43.11 | **44.72 (+3.73%)** | 44.63 (+3.53%) | 25.96 | 25.95 (-0.04%) | **28.97 (+11.59%)** | 28.91 (+11.36%) |
| BBH | 54.55 | 55.08 (+0.97%) | **55.81 (+2.31%)** | 42.70 | 40.68 (-4.73%) | 44.78 (+4.87%) | **44.84 (+5.01%)** |
| ARC-Challenge | 52.80 | **54.70 (+3.60%)** | 54.10 (+2.46%) | 55.70 | 54.62 (-1.94%) | 58.70 (+5.39%) | **58.79 (+5.55%)** |
| HellaSwag | 79.52 | **80.54 (+1.28%)** | 80.44 (+1.16%) | 80.29 | 80.28 (-0.01%) | 83.30 (+3.75%) | **83.36 (+3.82%)** |
| IFEval | 76.17 | 76.18 (+0.01%) | **77.02 (+1.12%)** | 71.97 | 73.04 (+1.49%) | 74.66 (+3.74%) | **74.99 (+4.20%)** |
| Mean Relative Gain | - | +2.29% | **+3.25%** | - | -0.72% | +6.19% | **+6.96%** |
| Time per Step (Sec) | $34.44 \pm 3.32$ | $34.74 \pm 3.49$ | $34.64 \pm 3.44$ | $43.01 \pm 2.74$ | $43.98 \pm 3.3$ | $44.22 \pm 3.34$ | $44.01 \pm 3.43$ |

## C Theoretical Analysis

### C.1 OTB Formulation

Following Li et al. (2026b), let's define some notations.

- $x$ be the input prompt,
- $y_{i,t}$ be the token generated by the $i^{\text{th}}$ rollout at time step $t$ and $y_{i,<t}$ be the preceding partial response,
- $\mathcal{G}$ be the indices of rollouts for $x$,
- $r_i$ be the outcome reward for $i^{\text{th}}$ rollout,
- $L$ be the maximum response length observed in the group,
- $T_i$ be the length of $i^{\text{th}}$ rollout,
- $m_{i,t} = \mathbb{I}[t \leq T_i]$ be the mask for $i^{\text{th}}$ rollout at time step $t$,
- $r_{i,t} = r_i \mathbb{I}[t = T_i]$ be the reward for $i^{\text{th}}$ rollout at time step $t$,
- $S_i = \sum_{u=1}^{L} m_{i,u} r_{i,u} = r_i$ be the cumulative reward for $i^{\text{th}}$ rollout,
- $G_{i,t} = \sum_{u=t}^{L} m_{i,u} r_{i,u} = m_{i,t} r_i$ be the reward-to-go for $i^{\text{th}}$ rollout at time step $t$,
- $\omega_{i,t} = 1 - 2\pi_{\text{old}}(y_{i,t}|x, y_{i,<t}) + \|\boldsymbol{\pi}_{\text{old}}(\cdot \mid x, y_{i,<t})\|_2^2$ be the token-level logit gradient proxy for $i^{\text{th}}$ rollout at time step $t$,
- $W_{i,t} = \sum_{u=1}^{t} m_{i,u} \omega_{i,u}$ be the cumulative realized energy for $i^{\text{th}}$ rollout at time step $t$.

The OTB baseline and token-level advantage are defined as follows.

$$B_t^{\text{OTB}} = \frac{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t} G_{i,t}}{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t}} \tag{3}$$

$$= \frac{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t} r_i}{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t}}, \tag{4}$$

$$A_{i,t}^{\text{OTB}} = m_{i,t}(G_{i,t} - B_t^{\text{OTB}}) \tag{5}$$

$$= m_{i,t}(r_i - B_t^{\text{OTB}}). \tag{6}$$

For ODRPO variant, we define the ordinal levels and baseline as follows.

$$b_i^{(k)} = \mathbb{I}[S_i \geq k], \tag{7}$$

$$B_t^{(k),\text{OTB-ODRPO}} = \frac{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t} b_i^{(k)}}{\sum_{i\in\mathcal{G}} m_{i,t} W_{i,t}}. \tag{8}$$

Thus, the advantage for ODRPO variant is defined as follows.

$$A_{i,t}^{\text{OTB-ODRPO}} = m_{i,t} \sum_{k=1}^{K} w^{(k)} \frac{b_i^{(k)} - B_t^{(k),\text{OTB-ODRPO}}}{\sigma_t^{(k)}}, \tag{9}$$

where $\sigma_t^{(k)} = \sqrt{B_t^{(k),\text{OTB-ODRPO}}(1 - B_t^{(k),\text{OTB-ODRPO}})}$ is the OTB-weighted standard deviation of the Bernoulli variable $b_i^{(k)}$ and $w^{(k)}$ is the weight for $k^{\text{th}}$ ordinal level, which will be $w^{(k)} = \sqrt{k}(0.1 + 4\mu^{(k)}(1-\mu^{(k)}))$, $\mu^{(k)} = \sum_{i\in\mathcal{G}} b_i^{(k)}/|\mathcal{G}|$ for Gini weighting. Independently standardizing each ordinal level preserves its distinct contribution to the final advantage, whereas without per-level standardization, uniform accumulation reduces exactly to standard OTB on the original scalar reward.

For the normalized OTB variant, we standardize the OTB advantages using the mean and standard deviation computed jointly over all valid response tokens in the batch. The same valid-token standardization is applied after ordinal-level accumulation for the OTB-ODRPO variants.

### C.2 Continuous Extension of ODRPO

We assume, without loss of generality, that rewards are ordered $r_1 \leq r_2 \leq \cdots \leq r_G$. While we utilize this ordering for indexing, the formulation naturally generalizes to duplicated rewards. We define the $i^{\text{th}}$ reward and associated statistical quantities (Bernoulli) of the group for $x \in \mathbb{R}$ as follows.

$$r_i(x) = \mathbb{I}\{r_i \geq x\} - \mathbb{I}\{0 \geq x\}, \tag{10}$$

$$\mu(x) = \mathbb{E}_{i\in\mathcal{G}}[r_i(x)], \tag{11}$$

$$\sigma(x) = \sqrt{\mu(x)(1-\mu(x))}. \tag{12}$$

The reward decomposition satisfies $r_i = \int_{-\infty}^{\infty} r_i(x)\, dx$. Considering the fact that the rewards will be centered for advantage calculation, we will drop the second indicator for the reward function and have $r_i(x) = \mathbb{I}\{r_i \geq x\}$ (mean $\mu(x)$ will be adjusted accordingly). Note that this simplification doesn't change the formulation. Now, accumulating advantage for each $x \in \mathbb{R}$, we get

$$A_i = \int_{-\infty}^{\infty} \frac{r_i(x) - \mu(x)}{N(x)}\, dx \tag{13}$$

$$= \int_{R_1}^{R_G} \frac{r_i(x) - \mu(x)}{N(x)}\, dx. \tag{14}$$

The second equality is a consequence of two facts: (i) for $x < r_1$, $r_i(x) - \mu(x) = 0$ as all rewards in the group satisfy the threshold and thus $\forall i \in \mathcal{G}, r_i(x) = \mu(x) = 1$, so the lower limit of the integral can be raised to $r_1$ and (ii) for $x > r_G$, $r_i(x) - \mu(x) = 0$ as all rewards in the group fail the threshold and thus $\forall i \in \mathcal{G}, r_i(x) = \mu(x) = 0$, thus the upper limit of integral can be lowered to $r_G$.

Now, for finite group size $G$, the mean function is a step function defined as follows.

$$\mu(x) = \frac{G - k(x)}{G}, \tag{15}$$

where $k(x) = \min\{k \in \mathbb{N} : 1 \leq k < G \text{ and } r_k < x \leq r_{k+1}\}$. Considering that common normalizations for Bernoulli variable are dependent on the mean, for instance, standard deviation $N(x) = \sqrt{\mu(x)(1-\mu(x))}$ or MaxRL $N(x) = \mu(x)$ (Tajwar et al., 2026), the normalization function $N(x)$ is a step-wise function with steps aligning with that of the mean function. This implies that the integral in eq. (13) can be reduced to finite summation given as follows.

$$A_i = \sum_{k=1}^{i-1} \Delta_k \frac{1-\mu_k}{N_k} - \sum_{k=i}^{G-1} \Delta_k \frac{\mu_k}{N_k}, \tag{16}$$

where $\Delta_k = R_{k+1} - R_k$, $\mu_k = (G-k)/G$ and $N_k$ is the normalization value derived from $\mu_k$ (Standard Deviation $N_k = \sqrt{\mu_k(1-\mu_k)}$, MaxRL $N_k = \mu_k$).

Apply weighting $W_i(x)$ to have $(r_i(x) - \mu(x)) \cdot W_i(x)$ is equivalent to defining the reward spacing as $\Delta_{k,i} = U_i(R_{k+1}) - U_i(R_k)$, where $U_i(x) = \int W_i(x)\, dx$ is anti-derivative of $W_i(x)$.

### C.3 Curl Condition

Following Davis & Recht (2025), let's define some notations.

- $\mathcal{Q}$ be the question corpus (can also be thought of as question batch),
- $\mathcal{R} := \{R_1, \ldots, R_K\}$ be a reward space with $K$ discrete rewards,
- $R(x, y) \in \mathcal{R}$ be the reward for question-answer pair $(x, y)$,
- $C_k(x)$ is set of answers that receives reward $R_k$ for question $x$,
- $\pi_\theta$ is the policy model,
- $\mathbf{p} := (p_1, \ldots, p_K)$ be the probability vector where $p_k = \sum_{y \in C_k(x)} \pi_\theta(y|x)$,
- $M$ is the group size with reward distribution $\{r_1, \ldots, r_M\}$,
- $\mathbf{s}_i := (s_1^i, \ldots, s_K^i)$ be the leave-one-out statistics vector where $s_k^i = \sum_{j \neq i} \mathbb{I}_k(r_j)$ and $\mathbb{I}_k(r) = 1$ if $r = R_k$,
- $Z_i = \sum_{k=1}^K \mathbb{I}_k(r_i) f_k(\mathbf{s}_i)$ is the advantage value of $i^{\text{th}}$ rollout and $f_k$ is the advantage function that defines the advantage value for reward $R_k$ provided the leave-one-out statistics vector.

So, the conditional expectation of the update for a single question $x$ is as follows.

$$\mathbb{E}_{y_i \sim \pi(\cdot|x)} \left[ Z_i \nabla_\theta \log \pi_\theta(y_i|x) | \mathbf{S}_i = \mathbf{s} \right] = \sum_y g_{\mathbf{s}}(y) \nabla_\theta \pi_\theta(y|x) \tag{17}$$

$$= \sum_{k=1}^K \sum_{y \in C_k(x)} g_{\mathbf{s}}(y) \nabla_\theta \pi_\theta(y|x) \tag{18}$$

$$= \sum_{k=1}^K f_k(\mathbf{s}) \sum_{y \in C_k(x)} \nabla_\theta \pi_\theta(y|x) \tag{19}$$

$$= \sum_{k=1}^K f_k(\mathbf{s}) \nabla_\theta p_k, \tag{20}$$

where $g_{\mathbf{s}}(y) = Z_i|_{\mathbf{S}_i = \mathbf{s}}$. In the third equality, we use the fact that as $y \in C_k(x)$, the associated reward is $R_k$ and so $g_{\mathbf{s}}(y) = Z_i|_{\mathbf{S}_i = \mathbf{s}} = f_k(\mathbf{s})$. For unconditional expectation, we will need to

average over multinomial distribution $\mathbf{s}_i \sim \text{Multi}(M-1, \mathbf{p})$.

$$\mathbb{E}_{y_i \sim \pi(\cdot|x)} \left[ Z_i \nabla_\theta \log \pi_\theta(y_i|x) \right] = \sum_{k=1}^{K} \mathbb{E}_{\mathbf{s} \sim \text{Multi}(M-1,\mathbf{p})} \left[ f_k(\mathbf{s}) \right] \nabla_\theta p_k. \tag{21}$$

Now, let the objective and its gradient be as follows.

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{Q}} \left[ h(\mathbf{p}) \right], \tag{22}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim \mathcal{Q}} \left[ \sum_{k=1}^{K} \frac{\partial h}{\partial p_k} \nabla_\theta p_k \right]. \tag{23}$$

Matching the terms of eq. (21) and eq. (22), we get

$$\frac{\partial h}{\partial p_k} = \mathbb{E}_{\mathbf{s} \sim \text{Multi}(M-1,\mathbf{p})} \left[ f_k(\mathbf{s}) \right]. \tag{24}$$

For the curl analysis, let's define additional notations for multinomial expansion.

$$\mathcal{S}_n := \{ \mathbf{s} = (s_1, \ldots, s_K) : \forall 1 \le i \le K, s_i \ge 0 \text{ and } \sum_{i=1}^{K} s_i = n \}, \tag{25}$$

$$B_{\mathbf{s}}^n(\mathbf{p}) := n! \prod_{i=1}^{K} \frac{p_i^{s_i}}{s_i!} \tag{26}$$

$$= n! \frac{\left(1 - \sum_{i=1}^{K-1} p_i\right)^{s_K}}{s_K!} \prod_{i=1}^{K-1} \frac{p_i^{s_i}}{s_i!}, \tag{27}$$

$$\frac{\partial B_{\mathbf{s}}^n(\mathbf{p})}{\partial p_i} = n(B_{\mathbf{s}-\mathbf{e}_i}^{n-1}(\mathbf{p}) - B_{\mathbf{s}-\mathbf{e}_K}^{n-1}(\mathbf{p})). \tag{28}$$

So, the expectation of $f_k(\mathbf{s})$ under multinomial distribution given in eq. (24) is

$$\mathbb{E}_{\mathbf{s} \sim \text{Multi}(M-1,\mathbf{p})} \left[ f_k(\mathbf{s}) \right] = \sum_{\mathbf{s} \in \mathcal{S}_{M-1}} f_k(\mathbf{s}) B_{\mathbf{s}}^{M-1}(\mathbf{p}). \tag{29}$$

As the intrinsic dependency is on just $K-1$ probabilities ($\because p_K = 1 - \sum_{k=1}^{K-1} p_k$), we can define the shifted update field,

$$F_k := \mathbb{E}_{\mathbf{s} \sim \text{Multi}(M-1,\mathbf{p})} \left[ f_k(\mathbf{s}) - f_K(\mathbf{s}) \right] \tag{30}$$

$$= \sum_{\mathbf{s} \in \mathcal{S}_{M-1}} (f_k(\mathbf{s}) - f_K(\mathbf{s})) B_{\mathbf{s}}^{M-1}(\mathbf{p}). \tag{31}$$

The necessary and sufficient condition (as the domain is simplex it is simply connected) for the update field to be gradient of a scalar objective is that for all $1 \le i, j < K$

$$\frac{\partial F_i}{\partial p_j} = \frac{\partial F_j}{\partial p_i}, \tag{32}$$

which gives

$$\sum_{\mathbf{s} \in \mathcal{S}_{M-1}} \Big[ (f_i(\mathbf{s}) - f_K(\mathbf{s})) B_{\mathbf{s}-\mathbf{e}_j}^{M-2}(\mathbf{p}) - (f_j(\mathbf{s}) - f_K(\mathbf{s})) B_{\mathbf{s}-\mathbf{e}_i}^{M-2}(\mathbf{p}) - (f_i(\mathbf{s}) - f_j(\mathbf{s})) B_{\mathbf{s}-\mathbf{e}_K}^{M-2}(\mathbf{p}) \Big] = 0, \tag{33}$$

that finally after variable substitution and collecting terms becomes

$$\forall 1 \le i, j \le K, \forall \mathbf{s} \in \mathcal{S}_{M-2}, \tag{34}$$

$$f_i(\mathbf{s} + \mathbf{e}_j) - f_i(\mathbf{s} + \mathbf{e}_K) + f_j(\mathbf{s} + \mathbf{e}_K) - f_j(\mathbf{s} + \mathbf{e}_i) + f_K(\mathbf{s} + \mathbf{e}_i) - f_K(\mathbf{s} + \mathbf{e}_j) = 0. \tag{35}$$

### C.3.1 CURL VIOLATION

Consider $M = 2$, so $\mathcal{S}_0 = \{(0, 0, 0)\}$. Consider three reward setting so $K = 3, R_k = k$. So, the mean and variance will be

$$\overline{R}(\mathbf{s}) = \sum_{i=1}^{K} s_i R_i / M, \tag{36}$$

$$\mathrm{Var}(\mathbf{s}) = \sum_{i=1}^{K} s_i (R_i - \overline{R}(\mathbf{s}))^2 / M. \tag{37}$$

Then the functions for GRPO and MaxRL are

$$f_i^{\mathrm{GRPO}}(\mathbf{s}) = \frac{R_i - \overline{R}(\mathbf{s} + \mathbf{e}_i)}{\sqrt{\mathrm{Var}(\mathbf{s} + \mathbf{e}_i)}}, \tag{38}$$

$$f_i^{\mathrm{MaxRL}}(\mathbf{s}) = \frac{R_i - \overline{R}(\mathbf{s} + \mathbf{e}_i)}{\overline{R}(\mathbf{s} + \mathbf{e}_i)}. \tag{39}$$

Finally, the curl of GRPO and MaxRL will be

$$\left[f_1^{\mathrm{GRPO}}(\mathbf{e}_2) - f_1^{\mathrm{GRPO}}(\mathbf{e}_3)\right] + \left[f_2^{\mathrm{GRPO}}(\mathbf{e}_3) - f_2^{\mathrm{GRPO}}(\mathbf{e}_1)\right] + \left[f_3^{\mathrm{GRPO}}(\mathbf{e}_1) - f_3^{\mathrm{GRPO}}(\mathbf{e}_2)\right] \tag{40}$$

$$= 2 \sum_{i=1}^{3} \mathrm{sign}(R_i - R_{i(\mathrm{mod}\ 3)+1}) \tag{41}$$

$$= -2 \tag{42}$$

$$\left[f_1^{\mathrm{MaxRL}}(\mathbf{e}_2) - f_1^{\mathrm{MaxRL}}(\mathbf{e}_3)\right] + \left[f_2^{\mathrm{MaxRL}}(\mathbf{e}_3) - f_2^{\mathrm{MaxRL}}(\mathbf{e}_1)\right] + \left[f_3^{\mathrm{MaxRL}}(\mathbf{e}_1) - f_3^{\mathrm{MaxRL}}(\mathbf{e}_2)\right] \tag{43}$$

$$= 2 \sum_{i=1}^{3} \frac{R_i - R_{i(\mathrm{mod}\ 3)+1}}{R_i + R_{i(\mathrm{mod}\ 3)+1}} \tag{44}$$

$$= -\frac{1}{15}. \tag{45}$$

Thus, for $M = 2$ and $K = 3, R_k = k$, GRPO and MaxRL do not satisfy the curl condition and thus can't admit a scalar objective.

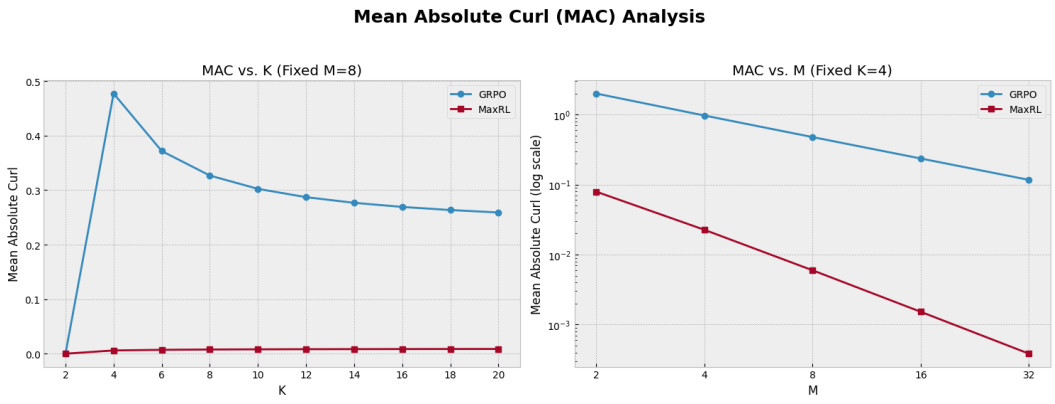



Figure 5: GRPO and MaxRL Mean Absolute Curl (MAC) value for varying $K$ and $M$.

As shown in fig. 5, both GRPO and MaxRL have non-zero MAC for increasing $K$. As group size $M$ increases, MAC decreases for both GRPO and MaxRL but as there is violation present, they still can't admit a pure scalar objective.

### C.4 ODRPO Objective Function

For arbitrary discrete reward space $\mathcal{R} := \{R_1, \ldots, R_K\}$, we will use the ODRPO advantage formulated in section C.2. Without loss of generality, we will impose ordering $R_1 < R_2 < \cdots < R_K$. The notations used in this section are defined as follows.

- $\Delta_m := R_m - R_{m-1}$ is the reward spacing.
- $S_m(\mathbf{s}) := \sum_{j=m}^{K} s_j$ is suffix sum or reverse cumulative sum.
- $t(s)$ is the ordinal advantage for success. For GRPO-like normalization, it will be $t(s) = \sqrt{\frac{M-(s+1)}{s+1}}$. $s+1$ is needed as we are working with leave-one-out statistics.
- $u(s)$ is the ordinal advantage for failure. For GRPO-like normalization, it will be $u(s) = -\sqrt{\frac{s}{M-s}}$.

Consequently, we can define the advantage function associated with each reward for ODRPO as below.

$$f_k(\mathbf{s}) = \sum_{m=2}^{k} \Delta_m t(S_m(\mathbf{s})) + \sum_{m=k+1}^{K} \Delta_m u(S_m(\mathbf{s})). \tag{46}$$

Using eq. (24) and eq. (46), we get

$$\frac{\partial h}{\partial p_k} = \sum_{m=2}^{k} \Delta_m \mathbb{E}_{\mathbf{s}\sim\text{Multi}(M-1,\mathbf{p})}[t(S_m(\mathbf{s}))] + \sum_{m=k+1}^{K} \Delta_m \mathbb{E}_{\mathbf{s}\sim\text{Multi}(M-1,\mathbf{p})}[u(S_m(\mathbf{s}))] \tag{47}$$

$$= \sum_{m=2}^{k} \Delta_m \mathbb{E}_{x\sim\text{Bin}(M-1,P_m)}[t(x)] + \sum_{m=k+1}^{K} \Delta_m \mathbb{E}_{x\sim\text{Bin}(M-1,P_m)}[u(x)] \tag{48}$$

$$= \sum_{m=2}^{k} \Delta_m \beta(P_m) + \sum_{m=k+1}^{K} \Delta_m \alpha(P_m) \tag{49}$$

$$= \sum_{m=2}^{k} \Delta_m (\beta(P_m) - \alpha(P_m)) + \sum_{m=2}^{K} \Delta_m \alpha(P_m), \tag{50}$$

where $P_m = \sum_{j=m}^{K} p_j$, $\beta(P_m) = \mathbb{E}_{x\sim\text{Bin}(M-1,P_m)}[t(x)]$, and $\alpha(P_m) = \mathbb{E}_{x\sim\text{Bin}(M-1,P_m)}[u(x)]$. The gradient of objective function becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{Q}}\left[\sum_{k=1}^{K} \frac{\partial h}{\partial p_k} \nabla_\theta p_k\right] \tag{51}$$

$$= \mathbb{E}_{x\sim\mathcal{Q}}\left[\sum_{k=1}^{K} \left(\sum_{m=2}^{k} \Delta_m (\beta(P_m) - \alpha(P_m))\right) \nabla_\theta p_k\right] \tag{52}$$

$$+ \mathbb{E}_{x\sim\mathcal{Q}}\left[\sum_{k=1}^{K} \sum_{m=2}^{K} \Delta_m \alpha(P_m) \nabla_\theta p_k\right]. \tag{53}$$

The term inside the second expectation becomes $C_\alpha \sum_{k=1}^{K} \nabla_\theta p_k = C_\alpha \nabla_\theta \sum_{k=1}^{K} p_k = C_\alpha \nabla_\theta 1 = 0$, where $C_\alpha = \sum_{m=2}^{K} \Delta_m \alpha(P_m)$. So, only the first expectation term survives.

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{Q}} \left[ \sum_{k=1}^{K} \left( \sum_{m=2}^{k} \Delta_m (\beta(P_m) - \alpha(P_m)) \right) \nabla_\theta p_k \right] \tag{54}$$

$$= \mathbb{E}_{x\sim\mathcal{Q}} \left[ \sum_{m=2}^{K} \Delta_m (\beta(P_m) - \alpha(P_m)) \left( \sum_{k=m}^{K} \nabla_\theta p_k \right) \right] \tag{55}$$

$$= \mathbb{E}_{x\sim\mathcal{Q}} \left[ \sum_{m=2}^{K} \Delta_m (\beta(P_m) - \alpha(P_m)) \nabla_\theta P_m \right]. \tag{56}$$

The last equality resembles the gradient of the objective function for binary reward. For GRPO, the objective function involves an arcsin term in the limit of large $M$ (Davis & Recht, 2025), so we will use that result directly for ODRPO GRPO's objective function.

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{Q}} \left[ \frac{2}{\pi} \sum_{m=2}^{K} \Delta_m \arcsin\left(\sqrt{P_m}\right) \right]. \tag{57}$$

For Gini weighting, as the weights are dependent on predefined level $k$ and ordinal bin's statistic, the objective function derivation remains the same as above just with different $\Delta_m$ values. Similar analysis holds for OTB case as the realized energy $W_{i,t}$ is computed independently from the reward $r_i$ and thus acts as a constant for the mini-batch.

### C.4.1 Effect of Batch Normalization

As the ODRPO advantage is computed by aggregating ordinal bin's advantage, the range of values depend on the reward granularity $K$. To contain the range of values, we can apply batch normalization to the advantage values. As group-wise advantage mean is zero, the effect of batch normalization is adaptive rescaling of the learning rate in optimizing the objective function eq. (54).

We tested the effect of batch normalization on ODRPO GRPO by comparing it against the unnormalized version on the GSM8K dataset (Cobbe et al., 2021) using Qwen2.5-3B-Instruct as the policy model and Qwen2.5-7B-Instruct (Qwen et al., 2025) as the auto-rater. For the unnormalized version, we divided the aggregated advantage by the number of ordinal bins $K$ to keep the range of values similar to that of the normalized version. The comparison is shown in fig. 6.

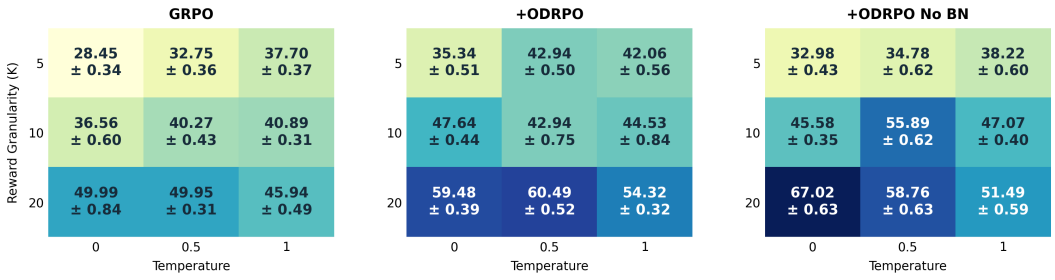



Figure 6: Effect of Batch Normalization across varying auto-rater's temperature ($T$) and reward granularity ($K$)

Across the $3 \times 3$ grid, ODRPO with batch normalization achieves a higher score in 6 of 9 settings than its unnormalized counterpart. Nevertheless, the unnormalized variant obtains a slightly higher mean score across the grid, $47.98$ versus $47.75$, a marginal difference of $0.23$ points. Both variants outperform GRPO in every setting, indicating that batch normalization has a limited effect on overall performance, while the primary gains are attributable to the ordinal decomposition.

### C.5 CURL VIOLATING WEIGHTS

Although ODRPO and its standard Gini-weighted variant satisfy the curl condition, enforcing this condition restricts the weight assigned to an ordinal bin to depend only on bin-level statistics. This limits the ability to adapt the weighting rule to the reward distribution of the sampled group. We therefore study three group-adaptive weighting rules Gini-Med, Gini-Mean, and Gini-Mean-Std defined as

$$w^{(k)}_{\text{Gini-Med}} = \sqrt{k} \cdot \left(0.1 + 4\mu^{(k)}(1-\mu^{(k)}) \cdot \exp\left(-\frac{\max(M_{\mathcal{G}} - k, 0)}{2}\right)\right), \tag{58}$$

$$w^{(k)}_{\text{Gini-Mean}} = \sqrt{k} \cdot \left(0.1 + 4\mu^{(k)}(1-\mu^{(k)}) \cdot \exp\left(-\frac{\max(\mu_{\mathcal{G}} - k, 0)}{2}\right)\right), \tag{59}$$

$$w^{(k)}_{\text{Gini-Mean-Std}} = \sqrt{k} \cdot \left(0.1 + 4\mu^{(k)}(1-\mu^{(k)}) \cdot \exp\left(-\frac{\max(\mu_{\mathcal{G}} - k, 0)}{2\sigma_{\mathcal{G}}}\right)\right). \tag{60}$$

Here, $M_{\mathcal{G}}$, $\mu_{\mathcal{G}}$, and $\sigma_{\mathcal{G}}$ are the median, mean, and standard deviation of the ordinal rewards in group $\mathcal{G}$. The exponential term suppresses bins below the selected group-level reference while retaining the full Gini factor for bins at or above it. The Gini-Mean-Std variant additionally normalizes this distance by the within-group standard deviation, making the weight sensitive to reward dispersion in a manner analogous to GRPO's standardized advantage.

Because these weights depend on statistics computed jointly from the sampled group, they need not define a conservative vector field. We quantify the resulting violation through (i) the fraction of curl components that are exactly zero and (ii) the MAC, reported both over all components and conditioned on the curl being nonzero. The results are shown in fig. 7.

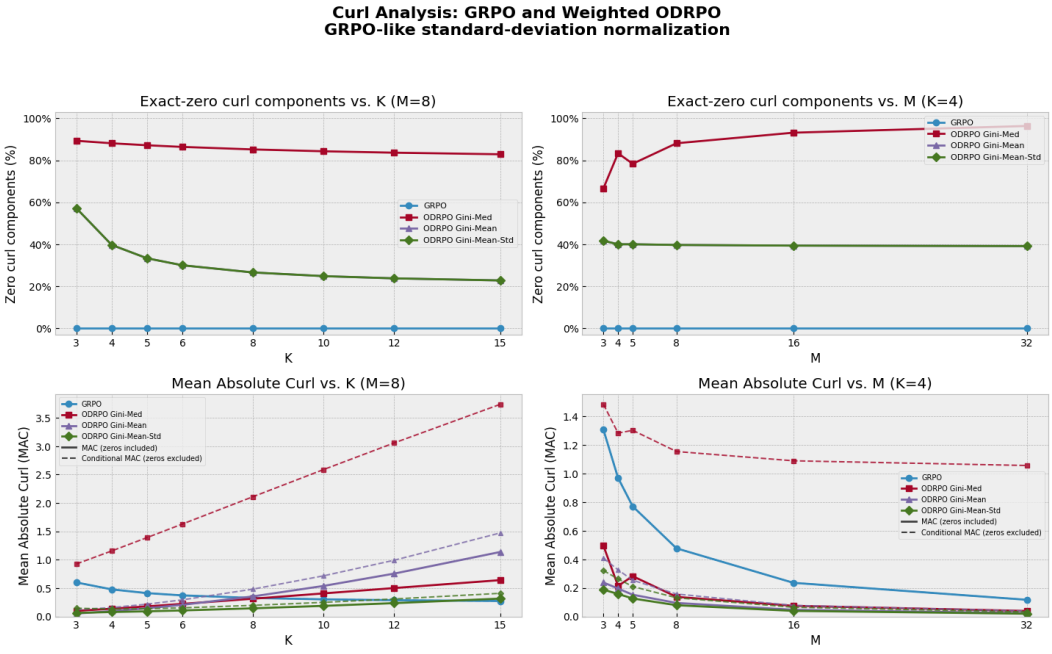



Figure 7: Curl Violation Analysis of Gini-Med, Gini-Mean, and Gini-Mean-Std.

For fixed $M = 8$, Gini-Med maintains approximately $83\%$ to $89\%$ exactly zero curl components, so its violation rate remains below $20\%$. The exact-zero fraction of Gini-Mean and Gini-Mean-Std decreases from roughly $57\%$ at $K = 3$ to $23\%$ at $K = 15$, indicating denser violations as reward granularity increases. For fixed $K = 4$, the exact-zero fraction of Gini-Med increases from approximately $67\%$ at $M = 3$ to $96\%$ at $M = 32$, whereas the mean-based variants stabilize near $40\%$. This difference is consistent with the median admitting only $\mathcal{O}(K)$ values, while the mean

can admit $\mathcal{O}(MK)$ values. The conditional MAC is larger than the overall MAC, particularly for Gini-Med, showing that its violations are sparse but can be locally sharp.

We evaluate these weights on GSM8K (Cobbe et al., 2021), using Qwen2.5-3B-Instruct as the policy and Qwen2.5-7B-Instruct (Qwen et al., 2025) as the auto-rater. Figure 8 reports the resulting accuracies.

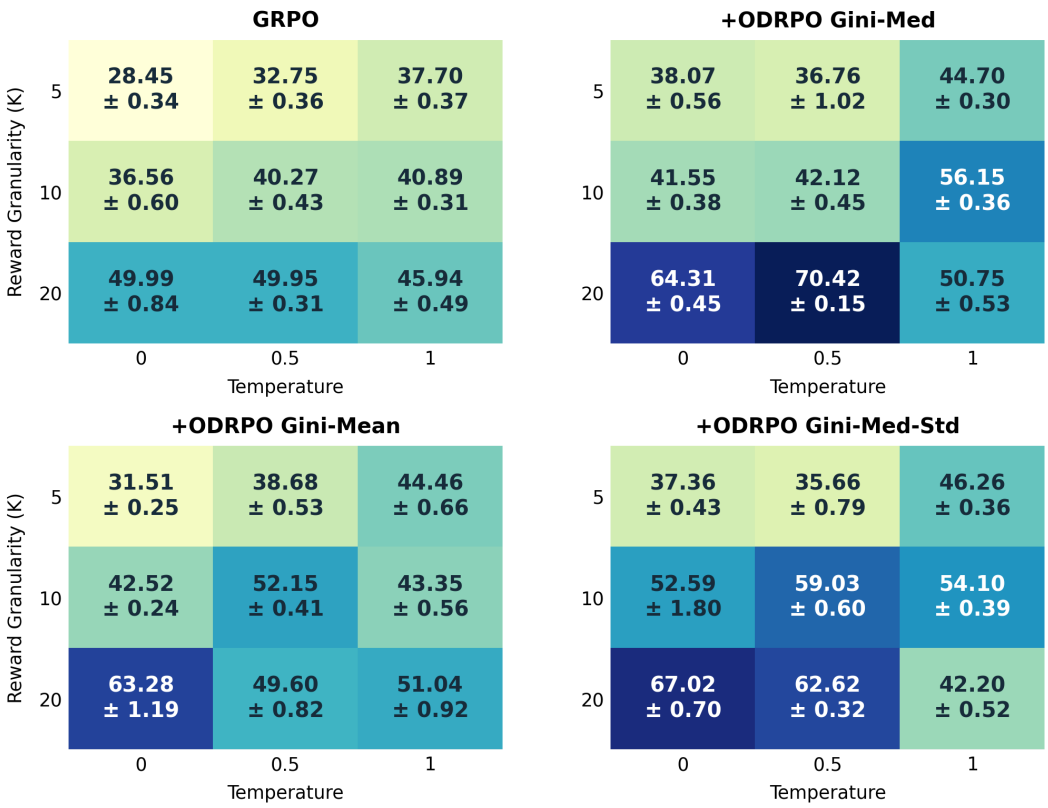



Figure 8: Effect of Curl-Violating Weights on ODRPO GRPO.

Gini-Med outperforms GRPO in all $9/9$ configurations, while Gini-Mean and Gini-Mean-Std each outperform it in $8/9$. Their mean accuracies are $49.43$, $46.29$, and $50.76$, respectively, compared with $40.28$ for GRPO. Among the adaptive variants, Gini-Med beats each alternative in $5/9$ configurations, while Gini-Mean-Std beats Gini-Mean in $7/9$ and achieves the highest mean accuracy.

These results show that group-dependent weighting can reintroduce curl violations without erasing the performance gains of the underlying ODRPO framework. Despite substantial differences in the frequency and magnitude of their violations, all three variants remain stronger than GRPO across the evaluated grid. This consistency indicates that the primary performance gain is driven by ODRPO, while the weighting rule controls how that gain is distributed across reward granularities and sampling temperatures.

## D Limitations and Future Work

Our experiments include both standard and reasoning-enabled auto-raters but focus on the black-box setting in which only a multi-tiered discrete reward is observed. Although ODRPO supports flexible weighting across ordinal thresholds, further work could develop information-theoretic, learnable, or annealed weighting schemes that adapt throughout training while preserving, or explicitly characterizing departures from, the global-objective guarantees established here. Beyond black-box LLM evaluation, future work should examine settings in which auto-rater internals, logits, or

confidence estimates are available, as well as feedback supplied by verifiers, human evaluators, or environment reward functions. Extending the empirical analysis to robotics or physical simulation, together with evaluating the continuous-reward extension of ODRPO, will help determine how broadly ordinal decomposition generalizes across reward sources, application domains, and arbitrary reward spaces.