The Peril of Preference: Why GRPO fails on Ordinal Rewards

Anisha Garg anisha.garg@cerebras.net
Ganesh Venkatesh ganesh.venkatesh@cerebras.net

Applied AI Research, Cerebras

Group-relative Policy Optimization's (GRPO) simplicity makes it highly desirable for adapting LLMs to become experts at specific tasks. But this simplicity also makes it ill-specified as we seek to enhance RL training with richer, non-binary feedback. When using ordinal rewards to give partial credit, GRPO's simplicity starts to hurt, as its group-average baseline often assigns a positive advantage to failed trajectories and reinforces incorrect behavior.
We introduce Correctness Relative Policy Optimization (CoRPO), a new formulation that solves this flaw. CoRPO uses an adaptive baseline that enforces a minimum quality threshold, ensuring failed solutions are never positively reinforced. Once the policy consistently meets this threshold, the baseline automatically transitions to a relative preference mode, pushing the model to find optimal solutions rather than just "acceptable" ones. We empirically validate CoRPO on a code verification task, where it demonstrates more stable convergence and better out-of-domain generalization.
This work represents a critical step in our broader research program to enable LLMs to learn genuinely new capabilities through reinforcement learning. We achieve this by enabling LLMs to learn from rich, multi-dimensional feedback - progressing from binary to ordinal rewards in this work, and onward to denser, per-step supervision.

1. Introduction

Group-relative Policy Optimization (GRPO) [1] has been widely adopted for finetuning Large Language Models (LLMs) on verifiable tasks such as mathematics and code generation [2–4]. Its popularity stems from its simplicity and efficiency, particularly using a group's average reward as a baseline instead of a complex value function. However, this simplification harbors a critical flaw when applied to tasks with ordinal rewards (e.g., 1–5 ratings) instead of binary preferences. In this paper, we mathematically demonstrate that GRPO's baseline is ill-specified for such tasks, and propose a robust formulation to address it.

The transition from PPO [5] to GRPO [1] in recent RL literature is primarily motivated by computational and memory efficiency. PPO relies on a value function, often a model of comparable size to the policy, which is computationally burdensome and can be difficult to train accurately in an LLM context. GRPO elegantly sidesteps this by obviating the need for a value function, instead using the average reward of a group of sampled outputs as the baseline [1]. However, we argue this simplification has a critical, implicit consequence: it fundamentally reframes the learning objective from one of absolute value to one of **relative preference**. The advantage is no longer measured against a learned, absolute expectation of reward, but against the performance of its sampled peers.

This poses a significant challenge. The GRPO authors rightly note that this "group relative" approach aligns well with the "comparative nature of rewards models," which are often trained on preference datasets [1]. This implicit preference-learning behavior is **severely amplified by the ordinal rewards** mentioned earlier. For the verifiable tasks where GRPO is increasingly used, the goal is to learn a signal for **correctness** (e.g., "is this solution correct?"), not **preference** (e.g., "which of these two solution is less wrong"). This mismatch is the root cause of the flaw we identify: GRPO's formulation can, and often does, assign a positive advantage to failed trajectories, actively reinforcing sub-optimal behavior simply because it is "better than average".

To solve this, we introduce **Correctness Relative Policy Optimization (CoRPO)**, a new advantage formulation. The main insight of CoRPO is to extend GRPO's simple central-tendency baseline to also provide a strict **Correctness Guarantee**. It achieves this with an adaptive baseline that prevents failed trajectories from ever receiving positive reinforcement, while still leveraging the group average to create an **Aspirational Drive** to continuously push the policy from "good" to "optimal" as it improves. This document details the flaw in GRPO's formulation and presents the mathematical framework for CoRPO.

Finally, we present an empirical validation of our approach. We apply CoRPO to a challenging task: training an RL model to verify the correctness of code generated by LLMs. Our results demonstrate that CoRPO provides more stable convergence and achieves higher final performance, validating its effectiveness in solving the practical issues inherent in GRPO's original design.

2. Challenge with the GRPO formulation for estimating baseline

2.1. Baseline and Advantage Calculation

Let G be the group size (i.e., the number of rollouts sampled from the policy). We have a group of G rollouts $\{y_1, y_2, ..., y_G\}$ sampled from the policy π_θ. The baseline b in GRPO is the average reward over this group:

$$b = \frac{1}{G}\sum_{i=1}^{G} R(y_i)$$

The Advantage $A(y_i)$ for each rollout y_i in the group is its reward shifted by this group baseline. Please note that we skip the normalization term $norm$ in the rest of the discussion for simplicity because our conclusions hold irrespective of the normalization approach [1, 6, 7].

$$A(y_i) = \frac{R(y_i) - b}{norm}$$

2.2. Per-Token Policy Gradient Loss

The simplified form of objective function to maximize, $J(\theta)$, is the expected advantage-weighted log-probability of the sequences in the group. Let T_i be the length of sequence y_i:

$$J(\theta) = \mathbb{E}_{y_i \sim \pi_\theta, i \in [1,G]}[\sum_{j=1}^{T_i} A(y_i) \cdot \log \pi_\theta(y_{i,j})]$$

This translates to the following minimization loss $\mathcal{L}(\theta)$, averaged over the group and per-token. *Note: past proposals also perform some form of token-length normalization as well which we skip for simplicity*:

$$\mathcal{L}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\left[A(y_i) \cdot \sum_{j=1}^{T_i} \log \pi_\theta(t_{i,j}|t_{i,<j})\right]$$

2.3. Challenge: Positive Advantage for a Failed Attempt

The critical flaw in this approach emerges when we analyze the conditions that produce a positive learning signal. Let's first define a **failed trajectory**, y_f, as any trajectory whose true, objective reward is negative. *Note: $R(y_f) < 0$ is not a requirement but a simplification for explanation without any loss of generality*:

$$R(y_f) < 0$$

Ideally, our RL algorithm should not update the policy to increase the probability of a failed trajectory. In policy gradient methods, this means we should always have $A(y_f) \leq 0$. However, the model's learning signal is based on the sign of the advantage, $A(y_i)$. The policy π_θ is updated to increase the probability of any sequence y_i for which:

$$A(y_i) > 0$$

The perverse learning signal where the model is encouraged to learn a failed trajectory y_f occurs when $A(y_f) > 0$. In the GRPO formulation, this occurs when:

$$R(y_f) - b > 0 \implies R(y_f) > b$$

The model will incorrectly reinforce a failed trajectory y_f (where $R(y_f) < 0$) as long as the group's average reward b is **even more negative** than $R(y_f)$. This creates a "positive advantage" for a failed trajectory whenever the following inequality holds:

$$b < R(y_f) < 0$$

Furthermore, we observe that this is not a rare edge case. It is the common scenario in complex problems where the policy is not yet well-trained for that task. If the policy generates a group of roll outs $\{y_1, ..., y_G\}$ that are all or mostly failures, the baseline b will be a large negative number.

In this state, any "less bad" failure y_f will satisfy $R(y_f) > b$, resulting in $A(y_f) > 0$. The per-token loss $\mathcal{L}(\theta)$ will then actively train the model to increase the probability of generating this objectively incorrect trajectory. The "better than average" logic, which is the cornerstone of GRPO's simplification, has reinforced a behavior we explicitly want to extinguish.

2.4. Heuristics from the community to address them

This fundamental flaw, where $b < R(y_f) < 0$, strongly suggests that GRPO's simple baseline is extremely sensitive to the data balance used to train the policy. This sensitivity may, in fact, be the underlying reason for several active research trends in the community. We can now see that techniques such as below are not just data-centric optimizations:

1. **Careful Data Curation:** Actively curating rollouts to maintain a "sweet spot" success rate (e.g., $0.2 < p_{success} < 0.8$) [8].
2. **Dynamic Rollout Selection:** Ensuring high variance in a batch though mixed trajectory types [9].
3. **Reward Re-weighting:** Biasing the training loss to focus more on negative examples or scaling down the influence of positive ones [10].

They can be seen as implicit, heuristic attempts to **manage the baseline** b. These methods are, in effect, workarounds to prevent b from becoming so negative that it assigns positive advantage to failed trajectories. This is a direct side-effect of this GRPO formulation when dealing with imbalanced, ordinal reward problems.

3. Correctness Relative Policy Optimization (CoRPO)

3.1. The Ideal Baseline: From Central Tendency to Quality

The flaw we just demonstrated where $b < R(y_f)$ reveals a fundamental mismatch in assumptions. In GRPO, the baseline b is designed to be a measure of **central tendency**. Its sole statistical purpose is to serve as an unbiased estimate for the value function, by using the policy's average performance, $\mathbb{E}[R(y)]$.

We argue that for LLM training, especially on challenging problems where the model's "tendencies" are mostly incorrect, this approximation is **insufficient**. When the average is a failure, using it as a baseline violates the goal of not reinforcing bad behavior. This leads us to a new premise: a baseline for LLM alignment should be a combination of measure of central tendency and of absolute quality. Based on this, we define an ideal baseline b_{ideal} as one that satisfies three core criteria:

- **Correctness Guarantee:** It must always discourage the model from learning sub-optimal habits. Mathematically, any failed trajectory y_f (with $R(y_f) < R_{correct}$) must never receive a positive advantage, providing a safety rail against reinforcing bad behavior, regardless of policy performance.

- **Proportional Feedback:** The negative feedback for a failed solution should be proportional to its quality. A solution that is "almost correct" should be penalized far less than one that is a catastrophic failure. This provides a smooth, informative gradient for the model to learn **how** to avoid failure.
- **Aspirational Drive:** Among solutions that are already "acceptable", the baseline must continue to provide a meaningful signal that encourages the model to improve, pushing it from a "good" solution to a "optimal" one. It must always incentivize progress toward the best possible outcome.

GRPO's formulation violates the **Correctness Guarantee** criterion and our CoRPO formulation presented in the next section is designed to meet all three.

3.2. Static Baseline: Correctness as a Threshold

Our first and most direct approach to satisfying our "ideal baseline" criteria is to replace GRPO's variable, central-tendency baseline with a **static, quality-based threshold**. Instead of comparing a reward to the group average, we compare it to a predefined constant, $R_{min\_correct}$, which represents the minimum reward for a trajectory to be considered "acceptably correct".

We define this new **static baseline** b_{static} as:

$$b_{static} = R_{min\_correct}$$

$$A_{static}(y) = R(y) - R_{min\_correct}$$

This change fundamentally alters the learning dynamic by addressing GRPO's primary flaw. We now evaluate this new advantage function against our three established criteria:

- **Correctness Guarantee:** This criterion is fully satisfied. By definition, any failed trajectory y_f has a reward $R(y_f) < R_{min\_correct}$. Therefore:

  $$A_{static}(y_f) = R(y_f) - R_{min\_correct} < 0$$

  The advantage for a failed solution is always negative, which eliminates the reinforcement of sub-optimal behavior.
- **Proportional Feedback:** This criterion is also met. The magnitude of the negative advantage, $|A_{static}(y_f)|$, is $R_{min\_correct} - R(y_f)$. This value is small when $R(y_f)$ is close to $R_{min\_correct}$ (a "near-miss") and large when $R(y_f)$ is far from $R_{min\_correct}$ (a severe failure). This provides a smooth, informative gradient for the model to learn from its failures.
- **Aspirational Drive:** This criterion reveals the new, critical trade-off. Consider a set of correct solutions, y_{good} and $y_{optimal}$, such that $R(y_{optimal}) > R(y_{good}) > R_{min\_correct}$.
  - For the good solution, $A_{static}(y_{good}) = R(y_{good}) - R_{min\_correct} > 0$.
  - For the optimal solution, $A_{static}(y_{optimal}) = R(y_{optimal}) - R_{min\_correct} > 0$.

  Both trajectories receive a positive learning signal encouraging the policy to learn both behaviors. As a result, this baseline provides very weak encouragement for $y_{optimal}$ over y_{good}. As the policy improves, it will continue to receive positive reinforcement for "merely acceptable" solutions, with no pressure to abandon them in favor of optimal ones. This formulation fails to incentivize the model to "aim higher" once it has reached the minimum bar for correctness.

This static baseline, while a significant improvement in teaching the policy how to solve the problem correctly, is overly conservative and fails to create a competitive "preference" among successful solutions. This motivates our final design, an adaptive baseline that seeks to integrate both safety and learning efficiency.

3.3. CoRPO: An Adaptive Correctness Baseline

The limitations of the static baseline motivate our final formulation for CoRPO, which is designed to be adaptive. We need a baseline that provides the correctness guarantees of b_{static} when the policy is performing poorly, but transitions to providing the aspirational drive of a relative preference baseline as the policy improves. We achieve this with a single, elegant function. First, we calculate the standard GRPO baseline (the group average):

$$b_{mean} = \frac{1}{G}\sum_{i=1}^{G} R(y_i)$$

Then, we define the CoRPO baseline b_{corpo} by "clamping" the baseline at the minimum correctness threshold:

$$b_{corpo} = \max(R_{min\_correct}, b_{mean})$$

The final CoRPO advantage $A_{corpo}(y_i)$ is then:

$$A_{corpo}(y_i) = R(y_i) - b_{corpo}$$

This adaptive formulation now satisfies all three of our ideal criteria by operating in two distinct phases:

Phase 1: Correctness-Seeking (When $b_{mean} < R_{min\_correct}$) : In this phase, the policy is performing poorly, so its average reward is below the bar for correctness. The baseline b_{corpo} is locked to $R_{min\_correct}$. The algorithm behaves identically to the **Static Baseline**. Correctness Guarantee is fully satisfied: Any y_f with $R(y_f) < R_{min\_correct}$ will have $A_{corpo}(y_f) < 0$.

Phase 2: Preference-Seeking (When $b_{mean} \geq R_{min\_correct}$) When the policy is performing well, the average reward (b_{mean}) is at or above the correctness threshold. The baseline b_{corpo} now becomes the standard GRPO baseline b_{mean}. As a result, the model is now pushed to prefer "optimal" solutions over "merely acceptable" ones, as they compete against each other relative to their new, higher average.

CoRPO thus provides a robust advantage function that seamlessly transitions from a "correctness" focus to a "preference" focus for different queries based on policy's capabilities.

4. Empirical Validation

In this section, we provide an empirical validation of our observations. Specifically, we apply our proposed baseline formulations to the task of RL training a model to verify correctness of code generated by LLMs. Our analysis first confirms the flaw in GRPO baseline estimation, then analyzes the training dynamics of our proposed CoRPO variants, and finally compares the downstream accuracy of all three methods.

4.1. Training Setup

We train an explanatory verifier for coding using reinforcement learning. Given a problem instance consisting of a question Q and two candidate responses (R_A, R_B), the policy outputs ratings $V = (v_A, v_B) \in [0, 10]$, representing the model's confidence in the correctness of each response. The reward is defined as the binary cross-entropy between the normalized difference of predicted ratings, $(v_B - v_A)$, and a target label of $\mathbb{I}(y_B > y_A)$, where $y_i \in \{0, 1\}$ is the ground truth label of corresponding response. This is an extension of the sum of BCE loss on individual ratings as described in [11].

The training dataset is curated from programming questions on CodeForces [12] and LeetCode [13]. For each question, we sample multiple solutions generated by Qwen3-8B to obtain both correct and incorrect reasoning trajectories. We remove content within the `<think>` tags and discard samples still exceeding 4,096 input tokens. To stabilize the absolute scale of model ratings, we also include a small subset of single-response judgment samples alongside the relative comparison pairs. In total, the final training set contains 4890 samples. We evaluate the model on three validation sets, each derived from tuples (Q, R_A, R_B)

- In-domain coding: one response correct and one incorrect (same format as training). Size: 196.

- Out-of-domain coding: both responses correct or incorrect. Size: 98.
- Out-of-domain math: one response correct and one incorrect. Size: 157.

We train Qwen3-8B using the GRPO, GRPO with static baseline, and CoRPO, following the VeRL [14] implementation. We train with a maximum sequence length (MSL) of 16,384, generating 8 rollouts per prompt with a global batch size of 512. We use dynamic filtering to discard rollouts with no variance and adhere to strictly on-policy training by performing a single gradient update for each batch of data generated. We set the KL coefficient and entropy coefficient to 0.0. Learning rate is set to 1×10^{-6} with a 10-step linear warmup, and rollouts done with temperature and top_p at 1.0.

4.2. Validation of the GRPO Baseline Flaw

We first empirically validate our central hypothesis from Section 2 that the standard GRPO baseline, b_{mean}, assigns a positive advantage to failed trajectories. To demonstrate this, we analyze a representative batch of rollouts (64 prompts rolled out 8 times) from an early training stage of our baseline GRPO model. At this point, the policy is not yet well-trained and produces many suboptimal solutions, creating the exact conditions for the $b < R(y_f) < 0$ flaw, where y_f is a failed trajectory.

Figure 1 visualizes this distribution. The chart clearly shows that a significant portion of the rollouts fall into the **Failed Trajectory, Positive Advantage** category. This empirically confirms that the standard GRPO formulation can reinforce sub-optimal behavior, violating the **Correctness Guarantee**. This finding motivates the need for our proposed baselines, designed to eliminate this pathological learning signal.

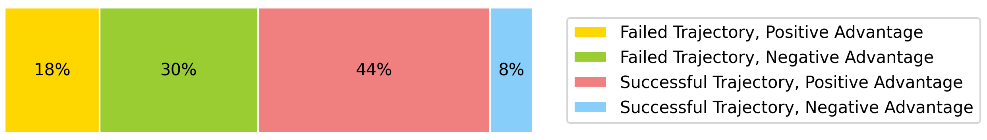

Figure 1: Distribution of advantage sign vs. trajectory status for a representative batch using GRPO baseline. The 18% slice for "Failed Trajectory, Positive Advantage" is empirical evidence of the $b < R(y_f) < 0$ flaw.

4.3. Analysis of Training Dynamics of CoRPO vs GRPO

This section analyzes the training dynamics of **GRPO** baseline, the **Static Baseline** (b_{static}), and our adaptive **CoRPO** baseline (b_{corpo}). Our analysis focuses on the nature of the feedback signal provided to the policy model throughout training. Figure 2 shows these trends for the baseline GRPO setting, our Static Correctness baseline (Static) and CoRPO proposal (Adaptive).

Initial Training Phase: The Correctness Guarantee First, we analyze the start of the training run, when the policy is not yet well-trained for the target task. We plot the ratio of positive to negative advantage signals ($r_{count} = \frac{\#\{A(y)>0\}}{\#\{A(y)<0\}}$) in Figure 2 (left), and ratio of their loss contributions ($r_{loss} = \frac{\Sigma Loss_{A(y)>0}}{\Sigma Loss_{A(y)<0}}$) in Figure 2 (right). At the beginning of training, both the **Static** and **CoRPO** baselines show r_{count} below 1.0. This indicates a much larger fraction of rollouts with negative feedback compared to GRPO.

This is the **Correctness Guarantee** in action. The GRPO baseline, by using group average, provides positive feedback to “less-bad” failures. In contrast, our proposed baselines correctly identify all failed trajectories ($R(y) < R_{min\_correct}$) and provides negative feedback required to discourage these behaviors.

Mid-to-Late Training Phase: Preference vs Optimal As the model’s performance improves, the limitations of the Static baseline become clear. As seen in Figure 2(right), the **Static** baseline’s ratio of positive-to-negative feedback rises dramatically as training progresses. This is because every “acceptably correct” trajectory ($R(y) \geq R_{min\_correct}$) receives a positive (or zero) advantage. As we argue in Section 3.3, this creates a very weak preference for “optimal” solutions over “merely acceptable” ones.

In contrast, the **CoRPO** baseline’s ratio stabilizes at a modest level. Once the policy is good enough to clear the $R_{min\_correct}$ threshold (i.e., $b_{mean} \geq R_{min\_correct}$), it transitions from a “correctness”model to a "preference" model. It correctly begins to assign negative advantages to "merely acceptable" solutions when superior solutions are present.

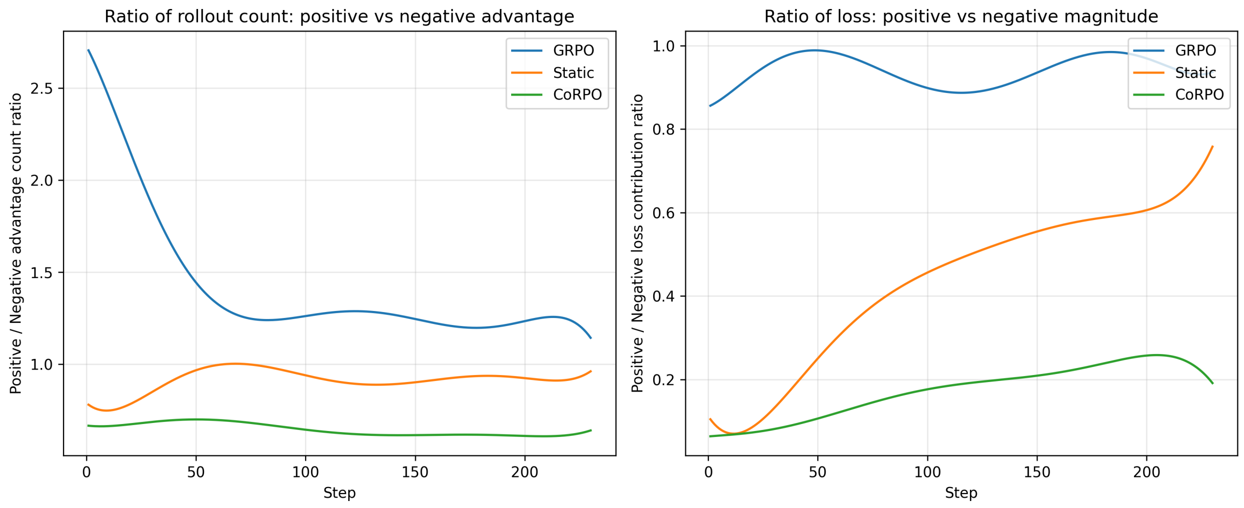

Figure 2: Ratio of Positive to Negative Signals over training steps for - GRPO baseline, Static Correctness baseline (Static) and CoRPO proposal. Left figure shows r_{count}, the ratio in terms of the rollout counts. Right figure plots r_{loss} which takes into account the advantage magnitude.

Impact on Learning Signal Magnitude While CoRPO is starting to demonstrate the desired training behavior, its conservative nature means the model takes smaller weight update steps. We observe that both in terms of the number of rollouts that lead to positive advantage as well as their loss contribution, CoRPO is significantly lower than GRPO and the static baseline. This slows down its learning for in-domain tasks (exploitation) but enables the model to continue to do exploration and perform well on out-of-domain tasks.

Beyond CoRPO, small advantage magnitude is a general challenge with ordinal rewards, where the weight update size can shrink if the policy predictions for the hard problems cluster close to each other instead of utilizing the full range. This behavior is significantly different from that of binary rewards, and it suggests a need for careful tuning of training hyperparameters, which we discuss in Section 6.

4.4. Downstream Accuracy

Finally, we compare downstream accuracy of models trained with each baseline. Table 1 shows the `pass@16` results for in-domain task of pairwise code verification, as well as two out-of-domain (OOD) tasks.

Static Baseline Performance As shown in Table 1, the **Static Baseline** (b_{static}) achieves mixed results on the in-domain tasks, performing worse on "First Correct" but better on "Second Correct" compared to GRPO. Its real strength, however, is revealed in the OOD evaluations. The Static baseline significantly outperforms GRPO on all four OOD tasks. This strongly suggests that by enforcing the **Correctness Guarantee** and preventing the model from learning from "less-bad" failures, the Static baseline learns a more robust and generalizable signal of true correctness, which translates to better OOD performance.

Adaptive CoRPO Performance We observe a similar, though slightly more nuanced, trend for our **CoRPO** baseline. It also demonstrates a clear improvement over GRPO on OOD tasks (e.g., +6.0 on "Both Incorrect" and +6.2 on "Both Correct"), confirming the benefit of the **Correctness Guarantee**. However, its overall upside is slightly lower than the Static baseline's on some tasks.

We hypothesize that this is a direct consequence of the training dynamics discussed in Section 4.3. The CoRPO baseline (b_{corpo}) and our training hyper-parameters results in a smaller average advantage magnitude (Figure 2). The current training hyperparameters and reward/advantage calculation have not been tuned for this, leading to smaller-than-optimal weight updates. Our current work, as discussed in

Section 6, focuses on adapting the training schedule and hyperparameters to ensure CoRPO can maintain consistent, impactful updates even as the model's capabilities for the task improves with training.

Table 1: Downstream accuracy comparison for GRPO baseline, our Static Correctness baseline (Static) and Adaptive CoRPO proposal. We analyze it both for in-domain task of code verification between a correct and incorrect response as well as two out-of-domain task. We report pass@16 for all the evaluations.

Task	**GRPO**	**Static (b_{static})**	**CoRPO(b_{corpo})**
	In-Domain Tasks		
First Correct	87.1	80.2	83.2
Second Correct	86.3	89.5	86.3
	Out-of-Domain Coding Tasks		
Both Incorrect	50.0	64.0	56.0
Both Correct	89.6	93.7	95.8
	Out-of-Domain Math Tasks		
First Correct	79.3	80.5	81.6
Second Correct	81.4	87.1	81.4

5. Conclusion

This work presents our current exploration of using Reinforcement Learning to robustly teach models new capabilities by providing feedback that is richer than a simple binary signal. We argue that the standard GRPO baseline, while efficient, is ill-specified for ordinal reward tasks and can lead to the reinforcement of sub-optimal behaviors.

Our central contribution, CoRPO, adjusts the baseline to enforce a strict **Correctness Guarantee**. As we have shown, this modification has demonstrated desirable properties, both in terms of correcting the pathological training dynamics of GRPO and in our initial downstream analysis, which shows improved generalization on out-of-domain tasks. This work surfaces new challenges in balancing correctness with learning efficiency, which we will address in our future work.

6. Discussion and Future Work

Our exploration has surfaced multiple issues we seek to address in the future. Our goal is to ensure RL goes beyond just biasing a model towards the most likely answer [15] and instead teach the model new capabilities by efficiently exploring the solution space. Our future work is focused on two primary directions.

Balancing Advantage Magnitude and Exploration A key challenge surfaced by our work is the inherent trade-off in using rich, ordinal rewards. As our analysis showed, this richness can lead to states where advantage magnitudes are very small when the policy predictions cluster closely together. This is a significant challenge for ordinal scales, unlike binary rewards where the update magnitude remains more consistent.

This directly impacts the exploration/exploitation balance. Our proposed CoRPO baseline, while safely avoiding bad habits, is conservative. This, combined with the small advantage magnitudes, results in the policy favoring exploration (improving OOD performance) at the cost of slower exploitation (less-dominant in-domain results). Our immediate future work is focused on solving this: we are exploring methods to ensure the model receives consistent, impactful updates, balancing its ability to learn correct behaviors with the drive to find optimal ones.

Beyond Outcome-Based Rewards While ordinal rewards are richer than binary ones, they are still a form of *outcome-based* reward, applied at the end of a trajectory. A significant future direction is to explore richer, denser feedback. This could involve per-step rewards that provide feedback *throughout* the generation, rather than just for the final state. This would provide a more fine-grained signal, enhancing the model's ability to learn complex, multi-step reasoning and problem-solving tasks more efficiently.

References

[1] 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.

[2] OpenAI. Openai o3 model documentation, 2025. URL https://platform.openai.com/docs/models/o3. Accessed: 2025-09-08.

[3] OpenAI. Introducing deep research, 2025. URL https://openai.com/index/introducing-deep-research/. Accessed: 2025-09-08.

[4] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

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

[6] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

[7] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

[8] Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning, 2025. URL https://arxiv.org/abs/2504.03380.

[9] Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. Treerpo: Tree relative policy optimization, 2025. URL https://arxiv.org/abs/2506.05183.

[10] Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning, 2025. URL https://arxiv.org/abs/2506.01347.

[11] Anisha Garg, Engin Tekin, Yash More, David Bick, Nishit Neema, and Ganesh Venkatesh. Calibrated reasoning: An explanatory verifier for dynamic and efficient problem-solving, 2025. URL https://arxiv.org/abs/2509.19681.

[12] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

[13] Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms, 2025. URL https://arxiv.org/abs/2504.14655.

[14] 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.

[15] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.