Extended Abstract

Motivation Large language models have shown remarkable success in complex reasoning tasks when fine-tuned via reinforcement learning (RL). However, applying online RL to combinatorial logic tasks such as the Countdown arithmetic game exposes a fundamental failure mode: *sparse reward collapse*. Early in training, most prompts are too difficult for a weakly initialized policy, causing all sampled completions to receive zero verifier reward. When this happens, group advantages collapse to zero, the policy gradient vanishes exactly, and training stalls completely. This is not a minor inconvenience, rather is a stable fixed point from which naive methods cannot escape. We hypothesize that overcoming sparse reward collapse requires two complementary inductive biases: a *performance-gated curriculum* that ensures the model always trains on problems it can occasionally solve, and *group-relative advantage normalization* (GRPO) that prevents the optimizer from destructively exploiting easy-stage rewards before harder problems are introduced.

Method We propose **GRPO+CRCLM**, a method that integrates GRPO with a multi-stage performance-gated curriculum. The training corpus is sorted by a two-level difficulty key (operand count first, target magnitude second) and partitioned into $K = 4$ cumulative stages. The model advances to the next stage only when its batch-mean verifier reward exceeds a threshold $\tau = 0.6$, guaranteeing a genuine learning foothold at each difficulty level before harder problems are introduced. GRPO normalizes rewards within each sampled group by their empirical mean and standard deviation, converting absolute reward values into relative within-group rankings and ensuring a well-scaled gradient signal regardless of batch difficulty. A PPO-style clipped surrogate objective prevents any single batch from inducing a destabilizing policy update during curriculum transitions.

Implementation We implement SFT, IPO, RLOO, and GRPO baselines on top of `Qwen2.5-0.5B`. All online RL methods are initialized from the same SFT checkpoint. We apply length-normalized sequence log-probabilities for importance weighting to prevent long responses from dominating gradient updates. The training pipeline uses vLLM for efficient rollout generation with group size $G = 8$ trajectories per prompt, and a binary-ternary verifier (1.0 for correct, 0.1 for correct format, 0.0 for incorrect) to score arithmetic reasoning traces. Training runs for 250 gradient steps on a single H100 GPU via Modal, tracked with Weights & Biases.

Results GRPO+CRCLM achieves a Pass@1 of **0.716** and Pass@16 of **0.860**, outperforming SFT, IPO, RLOO, and vanilla GRPO at every sample budget. Our method improves over vanilla GRPO by **+5.5%** at Pass@1, validating the independent contribution of curriculum gating. Vanilla GRPO itself outperforms RLOO by **+2.9%**, confirming that group-relative normalization provides a meaningful advantage over leave-one-out baselines under sparse reward conditions. Notably, applying the same curriculum to RLOO (RLOO+CRCLM) causes catastrophic degradation, placing it below even the SFT initialization, which is a direct consequence of RLOO's unclipped updates enabling reward hacking on the partial-credit formatting score.

Discussion Qualitative analysis of generation traces reveals the behavioral signature of this failure: RLOO+CRCLM learns to emit correctly formatted `<think>`/`<answer>` tags in indefinite loops, earning reliable partial credit without performing any arithmetic. GRPO avoids this via its $\epsilon_{\text{clip}} = 0.2$ bound and group-relative normalization, which together prevent tag emission from accumulating into a dominant policy feature. A key limitation is computational cost: generating $G = 8$ rollouts per step is substantially more expensive than SFT or IPO. The difficulty heuristic is also hand-designed, and the advancement threshold τ is fixed by inspection without a hyperparameter sweep.

Conclusion Performance-gated stage advancement is a necessary complement to group-relative normalization for online RL on sparse-reward reasoning tasks. A curriculum without a stable optimizer amplifies reward hacking rather than suppressing it; a stable optimizer without a curriculum wastes its budget on dead batches. The combination breaks the deadlock, and because it requires no architectural changes or reward shaping, it is a drop-in training-time intervention applicable to any verifier-based RL setup.

Multi-Stage Curriculum GRPO for Countdown

Timothy Yu*
Department of Computer Science
Stanford University
tyu2105@stanford.edu

Yash Ranjith*
Department of Computer Science
Stanford University
yranjith@stanford.edu

Abstract

Verifier-based online reinforcement learning has emerged as a compelling approach for teaching language models to reason, but applying it to combinatorial tasks exposes a fundamental failure mode: when problems are hard enough that random generation almost never succeeds, all sampled completions receive zero reward, group advantages collapse to zero, and the policy receives no useful gradient signal. We call this *sparse reward collapse*, and it is the primary bottleneck when training on the Countdown arithmetic task, where a model must combine a set of integers using basic operations to hit an exact target. Our key insight is that two complementary inductive biases together resolve this failure: a *performance-gated curriculum* that keeps the model training on problems it can occasionally solve, and *group-relative advantage normalization* (GRPO) that prevents the optimizer from degenerately exploiting early-stage rewards. We propose GRPO+CRCLM, which sorts the training corpus by operand count and target magnitude, partitions it into cumulative stages, and advances the model only when its batch-mean verifier reward exceeds a threshold. In experiments on `Qwen2.5-0.5B`, our method achieves a Pass@1 of **0.716** and Pass@16 of **0.860**, outperforming vanilla GRPO by $+5.5\%$ and RLOO by $+8.4\%$ at Pass@1. Qualitative trace analysis further reveals that RLOO under curriculum training degenerates into pathological `<think>`/`<answer>` loops, while GRPO's clipped objective and group normalization maintain coherent, structured reasoning chains. Together, these results show that performance-gated stage advancement is an effective inductive bias for online RL on sparse-reward reasoning tasks.

1 Introduction

Language models have recently demonstrated a striking ability to solve complex reasoning problems; this is not merely by recalling answers from training data, but by generating multi-step chains of thought that mirror deliberate problem-solving (Wei et al., 2022; Kojima et al., 2022). A key driver of this capability has been reinforcement learning from verifiers: rather than imitating human demonstrations, a model learns by attempting problems, receiving automatic correctness feedback, and updating its policy toward better solutions. Systems trained this way on mathematics and code have achieved remarkable results (Shao et al., 2024; Lian et al., 2025; Cobbe et al., 2021; DeepSeek-AI et al., 2025), suggesting that online RL with programmatic reward may be a broadly applicable recipe for capable reasoning, if it can be made to work reliably.

The difficulty is that this recipe has a critical failure mode. Consider the Countdown arithmetic task: given a set of 3–4 integers, the model must construct a valid expression, using each number exactly once, that evaluates to a target value. For a randomly initialized or weakly trained policy, the probability of stumbling onto a correct solution through random generation is vanishingly small. When all G completions sampled for a prompt receive zero verifier reward, the group advantage collapses to zero for every sample; the policy gradient is exactly zero, and training stalls completely. We call this *sparse reward collapse*. It is not a minor inconvenience since it is a stable fixed point

*Equal contribution

from which RLOO-style methods cannot recover on their own (Andrychowicz et al., 2017). Naively sorting problems from easy to hard only partially helps, because RLOO's leave-one-out baseline does not sufficiently constrain variance. The optimizer aggressively overfits to the early-stage distribution and degenerates into pathological repeating generation loops.

Our key insight is that resolving sparse reward collapse requires two complementary inductive biases working together. A *performance-gated curriculum* ensures the model is always exposed to problems it can occasionally solve, maintaining the non-zero reward signal necessary for meaningful gradient updates. *Group-relative advantage normalization* (GRPO) prevents the optimizer from destructively exploiting those early successes: by normalizing rewards within a sampled group, advantages depend only on within-group solution quality rather than absolute reward scale, and a PPO-style clipped objective prevents aggressive policy updates when the curriculum distribution shifts.

We present **GRPO+CRCLM**, a method that integrates GRPO with a multi-stage performance-gated curriculum (Figure 1). The training corpus is sorted by a two-level difficulty key: operand count first, target magnitude second, and partitioned into K cumulative stages. The model advances to the next stage only when its batch-mean verifier reward surpasses a threshold τ, guaranteeing a learning foothold at every stage before harder problems are introduced. Importance weights use length-normalized sequence log-probabilities throughout, preventing long generations from dominating gradient updates. No architectural changes are required; the curriculum is a training-time intervention only. **We therefore investigate the following hypothesis: combining a performance-gated curriculum with group-relative advantage normalization (GRPO) can overcome sparse reward collapse and achieve reliable learning on combinatorial reasoning tasks where naive RL methods fail.**

In summary, our main contribution is GRPO+CRCLM, a recipe for combining performance-gated curriculum learning with GRPO to overcome sparse reward collapse in combinatorial reasoning tasks. Evaluated on Countdown with a `Qwen2.5-0.5B` backbone, GRPO+CRCLM achieves a Pass@1 of **0.716** and a Pass@16 of **0.860**, outperforming SFT, IPO, RLOO, and vanilla GRPO at every sample budget. Our method improves over vanilla GRPO by **+5.5%** at Pass@1, and vanilla GRPO itself outperforms RLOO by **+2.9%**, validating both components independently. Qualitative trace analysis further reveals that RLOO under curriculum training degenerates into repetitive `<think>`/`<answer>` loops, while GRPO consistently produces structured, concise, and correct reasoning chains.

2 Related Work

Supervised fine-tuning and preference optimization. Supervised Fine-Tuning (SFT) remains the standard warm-start for language model alignment, training policies via next-token cross-entropy on gold demonstration traces (Howard and Ruder, 2018). To go beyond static data, offline preference optimization methods such as DPO and IPO (Garg et al., 2025) optimize directly from datasets of chosen and rejected responses without an explicit reward model, sidestepping the instability of online RL (Garg et al., 2025; Rafailov et al., 2023). These approaches are sample-efficient and straightforward to implement, but are fundamentally limited by the coverage of their offline datasets: a model cannot improve beyond the quality of its demonstration or preference data. **GRPO+CRCLM instead operates fully online, continuously generating and scoring new rollouts against a verifier, allowing the policy to discover arithmetic strategies that are absent from any fixed offline corpus. (Bansal et al., 2026)**

Online policy-gradient methods. Online reinforcement learning methods score live rollouts with a verifier and update the policy via gradient estimates. REINFORCE and its variants (Schulman et al., 2017) reduce gradient variance through baselines; REINFORCE Leave-One-Out (RLOO) specifically uses the mean reward of the remaining $K - 1$ samples as the baseline for the K-th completion (Kool et al., 2019; Ahmadian et al., 2024). Group Relative Policy Optimization (GRPO) (Shao et al., 2024) extends this by normalizing rewards within a sampled group using their mean and standard deviation, computing relative advantages that depend only on within-group solution quality. Paired with a PPO-style clipped surrogate objective and a KL-divergence penalty against the reference policy (Stiennon et al., 2020), GRPO eliminates the need for a separate value network while providing substantially more stable updates than RLOO. **Our work adopts GRPO as the core optimizer and demonstrates that its grouped normalization is critical for stable training under a curriculum, where RLOO degenerates into pathological repetitive generation.**

Curriculum learning for language models. Curriculum learning, organizing training from easy to hard examples, has a long history in machine learning as a technique for avoiding sparse-reward or high-loss failure modes in the early stages of optimization (Bengio et al., 2009; Graves et al., 2017). Recent work has applied this idea directly to LLM reasoning: Parashar et al. (2025) show that sorting training problems by difficulty and training in an easy-to-hard sequence meaningfully improves performance on multi-step reasoning benchmarks by bypassing the zero-success plateaus that standard online RL encounters. A key design choice across these methods is the *advancement criterion*: prior work typically advances based on step counts or fixed epoch boundaries, independent of whether the model has actually learned the current stage. **GRPO+CRCLM instead gates stage advancement on a live batch-mean verifier reward threshold, ensuring the model secures a reliable, non-degenerate reward signal before being exposed to harder problems.**

Verifier-based reasoning and the Countdown task. The use of rule-based verifiers for online RL has proven highly effective for mathematical and logical reasoning, where correctness can be checked automatically without human annotation (Lian et al., 2025; Shao et al., 2024). The Countdown arithmetic task, in which a model must produce an expression using a given set of integers to reach a target value, has emerged as a standard benchmark for this setting due to its combinatorial difficulty and exact verifiability. Prior work applying online RL directly to Countdown and similar tasks frequently encounters the sparse reward collapse problem: when problems are too hard relative to the current policy, all sampled completions fail, the group advantage vanishes, and training stalls. **To our knowledge, GRPO+CRCLM is the first method to directly target this failure mode on Countdown through performance-gated curriculum advancement, rather than through architectural changes or reward shaping.**

3 Method

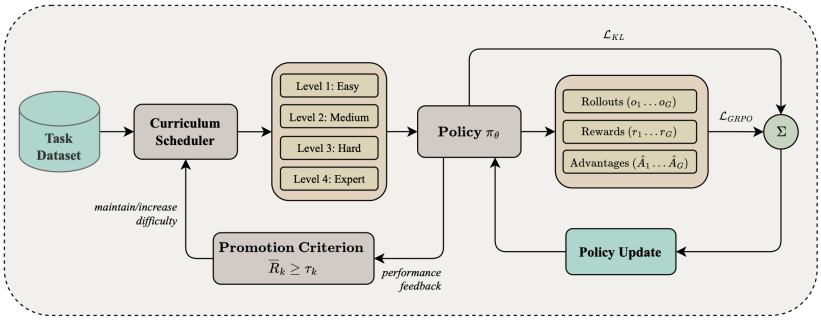

Figure 1: Our framework replaces repeated random exploration with a structured, performance-gated training trajectory. The pipeline consists of two stages: advantage estimation via Group Relative Policy Optimization (GRPO) and dataset progression via a performance-gated difficulty curriculum. All policy updates are applied to an initial Supervised Fine-Tuning (SFT) warm-start model.

3.1 Background: Verifier-Based Policy Gradients

In verifier-based reinforcement learning, we assume access to a dataset of prompts $\mathcal{D}$ and a deterministic, programmatic reward function $R(x, y)$ that evaluates a generated reasoning sequence y for a given prompt x. Standard policy gradient methods parameterize a policy π_θ and aim to maximize the expected reward $\mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}[R(x, y)]$. However, naive REINFORCE exhibits prohibitively high variance due to the sparse, binary nature of programmatic verifiers in combinatorial logic tasks. Baseline subtraction methods, such as REINFORCE Leave-One-Out (RLOO), mitigate this by using the mean reward of alternative samples as a baseline for advantage estimation. While effective in dense reward settings, we show that RLOO remains prone to pathological instability when reward scales fluctuate drastically under curriculum constraints.

3.2 Stage 1: Group Relative Policy Optimization (GRPO)

To stabilize policy updates without introducing the substantial computational overhead of a separate critic or value network, we employ GRPO. For each prompt x, the current policy π_θ autoregressively

samples a group of G distinct completions $\{y_1, \ldots, y_G\}$. We score each completion using the Countdown verifier:

$$r_i = R(x, y_i) \in \{0,\ 0.1,\ 1.0\} \tag{1}$$

where 1.0 indicates a mathematically correct response, 0.1 indicates correct formatting (the presence of `<answer>` tags) but an incorrect final value, and 0.0 indicates total failure. We normalize these raw rewards using the sample mean and standard deviation within the group to construct the advantage A_i:

$$A_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G) + \epsilon} \tag{2}$$

This group-level normalization converts absolute reward magnitudes into a relative within-group ranking. This ensures that the model receives a normalized gradient signal regardless of whether the absolute batch reward is near zero (early training) or near one (late training).

Because language models generate variable-length sequences, naive importance sampling ratios can explode or vanish for long rollouts. We compute the importance weights ρ_i using **length-normalized** sequence log-probabilities, preventing long trajectories from disproportionately dominating the gradient updates:

$$\bar{\ell}_\theta(y_i|x) = \frac{1}{L_i} \sum_{t=1}^{L_i} \log \pi_\theta(y_{i,t}|x, y_{i,<t}) \tag{3}$$

$$\rho_i = \exp\big(\bar{\ell}_\theta - \bar{\ell}_{\text{old}}\big) \tag{4}$$

The final policy gradient update minimizes a PPO-style clipped surrogate objective, regularized by an entropy bonus $H(\pi_\theta)$ to encourage diverse exploration, and a token-level KL divergence penalty against the frozen reference policy π_{ref} to anchor the optimization and prevent catastrophic forgetting:

$$\mathcal{L} = -\frac{1}{G} \sum_{i=1}^{G} \min\big(\rho_i A_i,\ \text{clip}(\rho_i,\ 1{-}\epsilon,\ 1{+}\epsilon)\, A_i\big) - \alpha H(\pi_\theta) + \beta\, \widehat{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \tag{5}$$

3.3 Stage 2: Performance-Gated Curriculum

Even with robust advantage estimation, GRPO suffers from sparse-reward collapse if the initial prompt distribution is too difficult. If all G samples yield $r_i = 0$, the variance is zero and $A_i = 0$ for all samples, resulting in a dead gradient.

To guarantee dense initial rewards, we dynamically control the difficulty of prompts x exposed to the model. We construct a two-level difficulty heuristic $d(x)$ for each Countdown prompt, prioritizing the number of operands first, followed by the magnitude of the target value:

$$d(x) = \big(|\text{numbers}(x)|,\ \text{target}(x)\big) \tag{6}$$

We sort the training corpus by $d(x)$ and define K cumulative stages at monotonically increasing fractional boundaries $f_1 < f_2 < \cdots < f_K = 1.0$. At stage k, the active training distribution is restricted to the easiest $\lfloor f_k N \rfloor$ examples.

Crucially, stage transitions are not tied to arbitrary step counts or epochs. The policy advances to stage $k+1$ strictly when its moving-average batch reward surpasses a hyperparameter threshold τ. This performance-gating mechanism functions as a strict algorithmic invariant: the model is guaranteed to have successfully established an exploratory foothold and learned robust logical primitives in the current distribution before it is forced to navigate the exponentially larger combinatorial spaces of harder prompts.

4 Experiments

Our experiments test whether GRPO+CRCLM, which combines performance-gated curriculum advancement with group-relative policy optimization, resolves sparse reward collapse and produces meaningful improvements in arithmetic reasoning on Countdown. We address four questions:

1. Does online RL with live verifier feedback improve over SFT and offline preference methods? If so, why can't offline methods keep up?

2. Does group-relative normalization (GRPO) produce more stable training than the leave-one-out baseline (RLOO) under the Countdown reward structure?
3. Does performance-gated curriculum advancement provide additional gains over vanilla GRPO, and at which sample budgets does it matter most?
4. What goes wrong when curriculum training is applied to RLOO specifically?

4.1 Experimental Setup

We evaluate all methods on the Countdown arithmetic task, where the model receives a set of 3–4 integers and must produce a valid expression, using each number exactly once, that evaluates to a specified target value. All experiments use `Qwen2.5-0.5B` as the backbone, so observed differences reflect algorithmic choice rather than model capacity. Every online RL run is initialized from the same SFT checkpoint, which is also frozen and used as the reference policy π_{ref} for KL penalties where applicable. Rollouts are generated with vLLM at group size $G = 8$ trajectories per prompt. All methods train for 250 gradient steps on the same training split of `asingh15/countdown_tasks_3to4`.

We report Pass@K for $K \in \{1, 2, 4, 8, 16\}$ on a disjoint held-out test set, where Pass@K measures the fraction of problems solved correctly by at least one of K sampled responses. At evaluation time we sample 16 responses per prompt (temperature 0.6, top-p 0.95, top-k 20) and compute Pass@k for $k < 16$ via the standard unbiased estimator from the saved pool. Pass@1 is our primary metric because it directly measures whether the model can produce a correct solution in a single attempt, which is the most demanding and practically relevant setting.

Baselines and ablations. We compare our full method against five configurations. **SFT** is the supervised fine-tuned checkpoint with no further training; it serves as the initialization for all online methods and as a lower bound on what online RL should surpass. **IPO** is offline preference optimization trained on a fixed set of chosen/rejected pairs (`asingh15/countdown_tasks_3to4-dpo`); it tests whether offline data alone can substitute for live exploration. **RLOO** is online policy gradient with a leave-one-out baseline, trained on the full dataset without any curriculum; it isolates the contribution of online rollout generation relative to IPO. **GRPO** is online policy gradient with group-relative reward normalization and PPO-style clipping, trained on the full dataset without any curriculum; it isolates GRPO's normalization over RLOO. **RLOO+CRCLM** substitutes RLOO as the update algorithm inside the same performance-gated curriculum framework; it tests whether curriculum advancement alone suffices, or whether GRPO is also a necessary ingredient. Our full proposed method is **GRPO+CRCLM**.

4.2 Does Online RL Improve over SFT and Offline Methods?

We first ask whether live verifier feedback provides any meaningful advantage over static training signals. Both SFT and IPO hit hard ceilings that every online method surpasses by a substantial margin, as visible in Figure 2. The SFT ceiling arises from data coverage: the model can only recombine arithmetic strategies present in its demonstration corpus, and any solution path not represented there will never appear at test time. IPO offers only a limited escape. Because it optimizes over a static set of chosen/rejected pairs, it is similarly bounded by the coverage of that dataset, and its pairwise objective evaluates two complete responses holistically, providing no mechanism to localize *which step* in a multi-turn arithmetic chain was wrong, making credit assignment coarse for the kind of sequential reasoning Countdown demands.

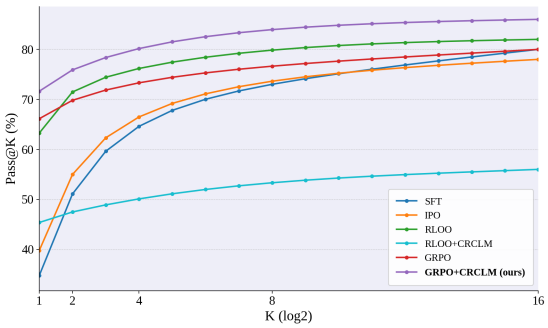

Figure 2: Pass@K for all methods at $K \in \{1, 2, 4, 8, 16\}$. GRPO+CRCLM leads at every budget; the gap is widest at K=1 and narrows as additional sampling compensates for individual failures.

Online methods address both issues directly. By generating rollouts from the live policy at each training step and scoring them with the Countdown verifier, RLOO and GRPO continuously explore solution paths that need not appear anywhere in the offline dataset. The verifier provides exact,

per-response correctness feedback tied to what the model actually produced, enabling fine-grained credit assignment that no static preference dataset can replicate. The gap between online and offline methods is largest at $K{=}1$, where individual generation quality matters most, and shrinks only modestly as K grows, indicating that online training produces genuinely better reasoning rather than merely lower variance.

4.3 Does Group-Relative Normalization Improve over RLOO?

Vanilla GRPO outperforms vanilla RLOO by $+2.9\%$ on Pass@1, a gap that holds across all values of K in Figure 2. This traces back to how each method constructs its advantage estimates under the sparse, discrete reward structure of the Countdown verifier.

RLOO computes the advantage for the i-th completion using the mean reward of the remaining $G-1$ samples as a baseline: $b_i = \frac{1}{G-1}\sum_{j\neq i} r_j$. In dense reward settings this baseline is informative, but the Countdown verifier assigns rewards from $\{0.0, 0.1, 1.0\}$, and many batches are effectively homogeneous: when the current policy finds a given prompt either systematically solvable or systematically unsolvable, all G rollouts fall into the same reward bin. In these homogeneous batches the leave-one-out baseline nearly matches each individual reward, leaving advantage estimates near-zero but noisy. As the model transitions between difficulty levels, the fraction of homogeneous batches and the scale of reward fluctuations change substantially from step to step, making the RLOO gradient signal unreliable precisely when stable updates are most needed.

GRPO standardizes within-group rewards by their empirical mean and standard deviation, $A_i = (r_i - \bar{r})/(\sigma_r + \epsilon)$, converting absolute values into a relative ranking that is always zero-mean regardless of batch difficulty. The PPO-style clipping additionally prevents any single high-advantage sample from inducing a destabilizing update, which is a property that is especially important during periods of rapid reward-distribution shift. Together, these two properties produce the more stable training trajectory that GRPO's $+2.9\%$ advantage over RLOO directly reflects.

4.4 Does Performance-Gated Curriculum Improve over Vanilla GRPO?

GRPO+CRCLM outperforms vanilla GRPO by $+5.5\%$ on Pass@1, reaching **0.716** Pass@1 and **0.860** Pass@16. This improvement demonstrates the independent value of the curriculum over and above the GRPO objective.

Without a curriculum, vanilla GRPO samples uniformly from the full training distribution at every step. In early training, harder problems (those with more operands and larger target values) almost never yield a correct rollout in any of the $G = 8$ completions. When all group rewards are zero, the group standard deviation $\sigma_r = 0$, every advantage A_i is identically zero, and the policy gradient vanishes exactly. These dead batches consume gradient steps without contributing any learning signal, wasting a significant fraction of the fixed 250-step training budget.

The curriculum eliminates dead batches during the cold-start period. By restricting the active training set to the easiest 25% of examples in stage 1, the model is initially exposed only to problems within its reach, ensuring that some fraction of rollouts per batch receive non-zero verifier reward. The advancement criterion, stage $k{+}1$ unlocks only when the batch-mean verifier reward exceeds $\tau = 0.6$, guarantees that the model has secured a genuine foothold at the current difficulty level before the training distribution expands.

As Figure 2 shows, GRPO+CRCLM's advantage over vanilla GRPO is most pronounced at Pass@1. This is precisely where the curriculum has the largest effect: by ensuring the model solves simpler problems reliably before advancing, it cultivates the consistent, convergent single-pass reasoning that Pass@1 directly measures. The gap narrows at Pass@16, as additional sampling compensates for some of vanilla GRPO's inconsistency, but GRPO+CRCLM maintains a meaningful lead even there.

4.5 Why Does RLOO Degenerate Under Curriculum Training?

Applying the same curriculum to RLOO (RLOO+CRCLM) produces a catastrophic failure: rather than improving over vanilla RLOO, the curriculum causes substantial performance degradation, placing RLOO+CRCLM below even the SFT initialization in Figure 2. Figure 3 reveals the mechanism. Unlike a well-behaved learner whose training reward tracks genuine progress, RLOO+CRCLM

achieves a consistently *higher* batch-mean reward than vanilla RLOO throughout training. This is not a sign of faster learning, but indicative of reward hacking.

The curriculum's early stages expose the model exclusively to easy problems, and RLOO's unclipped, unnormalized updates are aggressive enough to rapidly overfit to this narrow distribution. The model discovers that the verifier awards a partial score of 0.1 for any completion that contains a correctly formatted `<answer>` block, regardless of arithmetic correctness. Since the leave-one-out baseline is computed from the same group of shortcut-generating completions, the relative advantage of tag emission appears consistently positive, and RLOO's unclipped updates reinforce this behavior aggressively. The inflated training reward in Figure 3 is the direct result: the model earns reliable partial credit by emitting formatted tags, not by solving arithmetic.

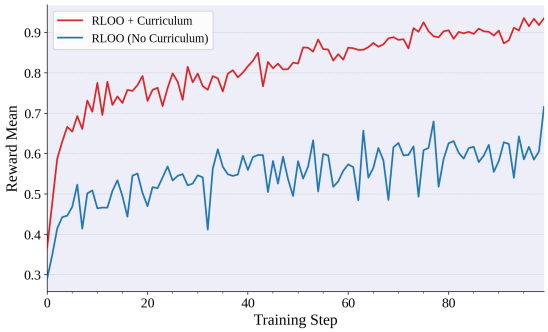

Figure 3: Batch-mean verifier reward over training for RLOO and RLOO+CRCLM. RLOO+CRCLM's higher training reward signals reward hacking, not faster learning; it does not transfer to held-out test performance.

The failure becomes apparent at test time. The shallow formatting pattern does not transfer since harder problems require genuine multi-step computation that tag emission cannot satisfy. Compounding this, when a generation reaches the maximum sequence length mid-loop the output is truncated, the final `<answer>` block is likely malformed, and the verifier assigns 0.0 to a trace that may have contained the correct answer earlier.

GRPO avoids this failure via two complementary properties. The $\epsilon_{\text{clip}} = 0.2$ bound on the importance weight ratio ρ_i prevents any single batch from shifting the policy far enough to lock in a reward-hacking shortcut. Group-relative normalization ensures that the advantage of tag emission is assessed against the full within-group reward distribution, rather than a leave-one-out baseline that may itself be dominated by the same shortcut. Together, these constraints prevent the inflated-reward equilibrium that RLOO+CRCLM settles into.

4.6 Qualitative Results

To make the behavioral difference concrete, we inspect generation traces from both policies on a held-out test example: reach target 98 using operands $[44, 19, 35]$. GRPO+CRCLM produces a single focused reasoning chain and terminates immediately upon finding a valid solution. RLOO+CRCLM finds the correct answer on the first attempt but then enters an indefinite loop, re-emitting `<think>`/`<answer>` pairs with no stopping condition, which is the direct generative signature of the reward-hacking described above. When such a generation reaches the maximum sequence length mid-loop, the final `<answer>` block is truncated and the verifier assigns 0.0 to a trace that contained the correct answer earlier.

GRPO+CRCLM (correct, terminates)

```
<think>
44 + 35 = 79
79 + 19 = 98.  Works!
</think>
<answer>(44 + 35) + 19</answer>
```

RLOO+CRCLM (correct, then loops)

```
<think>
44 + 19 = 63; 63 + 35 = 98!
</think>
<answer>(44 + 19) + 35</answer>
<think>
Verify:  63+35=98.  Perfect!
</think>
<answer>(44 + 19) + 35</answer>
<think>
...  [continues looping]
```

GRPO avoids this because its $\epsilon_{\text{clip}} = 0.2$ bound prevented tag emission from accumulating into a dominant policy feature during easy curriculum stages. Across all test traces inspected, GRPO+CRCLM outputs are uniformly structured: a single `<think>` block followed immediately by a single `<answer>` block. The proximity of its Pass@1 (0.716) and Pass@16 (0.860) reflects this directly since the majority of correct solutions arrive on the first sample, not recovered through additional attempts.

5 Discussion

Limitations. GRPO+CRCLM carries several practical constraints worth acknowledging. The most immediate is computational: generating $G = 8$ rollouts per prompt per step, with the

SamplingWorker and GRPOUpdateWorker alternating GPU ownership via ray.kill, is substantially more expensive than SFT or IPO and limits the batch size we can sustain within a single H100. Our difficulty heuristic (sorting by $(\text{num\_operands}, \text{target})$) is also a hand-designed proxy that captures coarse structure but ignores combinatorial properties of specific number sets that make some problems harder than others at the same operand count. The advancement threshold $\tau = 0.6$ is a hyperparameter we fixed by inspection rather than sweeping, and sensitivity to this value is unexplored. Finally, all results are from a single training run on Qwen2.5-0.5B; we do not have error bars across seeds or evidence that the method scales to larger models without retuning.

Difficulties encountered. The central difficulty in this project was the sparse reward collapse itself. We originally proposed off-policy sampling algorithms, but early runs showed that the policy simply stalled when problems were too hard, producing flat reward curves and no learning signal. This forced the pivot to an online curriculum approach. A second difficulty was the instability of the Ray actor alternation: vLLM does not cleanly release GPU memory between inference and training passes, and we had to explicitly call ray.kill and force torch.cuda.empty_cache() to avoid OOM errors on the H100. The RLOO degeneration was also initially confusing because the training reward for RLOO+CRCLM looked *better* than vanilla RLOO throughout training, and only examining the test Pass@K and inspecting the raw generation traces revealed that this was reward hacking rather than genuine improvement.

Broader impact. Our core finding, that performance-gated stage advancement is a necessary complement to group-relative normalization for online RL on sparse-reward tasks, generalizes beyond the Countdown setting. Any domain where a verifier can check correctness but random generation almost never succeeds (formal theorem proving, code synthesis, symbolic manipulation) faces the same cold-start failure mode. GRPO+CRCLM provides a simple, architecture-free recipe for addressing it: sort by difficulty, gate stage transitions on live reward, and use a clipped group-normalized objective to prevent the optimizer from exploiting the easy stages. Our negative result, that applying curriculum to RLOO makes things worse, not better, is equally transferable as a practical warning: a curriculum without a stable optimizer amplifies reward hacking rather than suppressing it.

6 Conclusion

The central lesson of this work is that sparse reward collapse is not just a nuisance to be engineered around, rather it is a stable failure mode that actively resists naive fixes. A curriculum alone is insufficient: without a stable optimizer, it creates a new attack surface for reward hacking. A stable optimizer alone is insufficient: without structured exposure to solvable problems early in training, it wastes its budget on dead batches. The combination of performance-gated stage advancement with GRPO's clipped group normalization is what breaks the deadlock, and the +5.5% gain over vanilla GRPO and +2.9% gain of GRPO over RLOO reflect two independent, additive contributions rather than a single monolithic design choice.

The broader significance is that this recipe requires no architectural change, no reward shaping, and no human annotation beyond a difficulty ordering of the training set. It is a training-time intervention that can be dropped on top of any verifier-based RL setup. The negative result is equally significant: RLOO+CRCLM finishing below the SFT initialization it started from shows that optimizer choice is not interchangeable under a curriculum, and that practitioners who adopt easy-to-hard training without careful optimizer selection may make performance strictly worse.

Several concrete directions follow from this work. First, the difficulty heuristic is hand-designed; replacing it with a learned difficulty estimator, such as the current policy's own pass rate on each problem, would make the curriculum adaptive rather than fixed. Second, we disabled the KL penalty ($\beta_{\text{KL}} = 0.0$) throughout; incorporating a process reward model (PRM) that provides dense step-level feedback could relax the dependence on a binary verifier and further stabilize training on problems where the final answer alone is an unreliable credit assignment signal (Lightman et al., 2023). Third, all experiments used Qwen2.5-0.5B; verifying that the curriculum gating threshold τ and group size G transfer to larger models without retuning is an open empirical question with direct practical relevance.

7 Team Contributions

- **Timothy Yu:** Implemented the Supervised Fine-Tuning (SFT) baseline, developed the GRPO algorithm for our extension, designed the curriculum difficulty sorting logic, and managed the overall training infrastructure and vLLM integration.
- **Yash Ranjith:** Implemented the Preference Optimization (IPO) baseline, developed the Online Policy-Gradient (RLOO) algorithm, built the evaluation pipeline with the rule-based verifier, and implemented the Pass@k metric calculations and training plots.

Both authors contributed equally to debugging and all written deliverables (including the Project Proposal, IPO+RLOO Milestone Report, Final Paper, and Poster Presentation). There were no changes or deviations from the initial project proposal.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, et al. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv preprint arXiv:2402.14740* (2024).

Marcin Andrychowicz, Filip Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in Neural Information Processing Systems* 30 (2017).

Rachit Bansal et al. 2026. RL Excursions during Pretraining. *arXiv preprint* (2026).

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

DeepSeek-AI et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948* (2025).

Ishaan Garg et al. 2025. IPO: Your Language Model is Secretly a Preference Classifier. *arXiv preprint arXiv:2502.16182* (2025).

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *International conference on machine learning* (2017), 1311–1320.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-Tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems* 35 (2022), 22199–22213.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Buy 4 REINFORCE samples, get a baseline for free!. In *International Conference on Learning Representations*.

Jiao Lian et al. 2025. Reinforcement Learning is all you need. *arXiv preprint arXiv:2503.09512* (2025).

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050* (2023).

Shubham Parashar et al. 2025. Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning. *arXiv preprint arXiv:2506.06632* (2025).

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Harrison Edwards, Ryan Jackson, William Saunders, Pamela Mishkin, Bill Clark, Amanda Askell, et al. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

A Implementation Details

Training infrastructure. All experiments are run on a single NVIDIA H100 GPU via Modal, with 8 CPU cores and a 24-hour timeout. All models are loaded in bfloat16 with gradient checkpointing enabled throughout. Checkpoints, optimizer states, and evaluation outputs are persisted to a Modal volume. Training runs are tracked with Weights & Biases.

Baselines. SFT fine-tunes `Qwen2.5-0.5B` via token-level cross-entropy masked to response tokens only, using AdamW (lr $= 5 \times 10^{-6}$, weight decay 0.01) with a cosine schedule and 5% linear warmup, gradient clipping at 1.0, max prompt length 512, and max response length 1024 tokens. IPO keeps a frozen deep-copy of the policy as a reference model and optimizes the pairwise objective $\mathcal{L}_{\text{IPO}} = \big((h - \frac{1}{2\beta})^2\big)$ where $h = \big(\log \frac{\pi(y_w)}{\pi_{\text{ref}}(y_w)} - \log \frac{\pi(y_l)}{\pi_{\text{ref}}(y_l)}\big)$, with $\beta = 0.1$ over a static paired dataset (`asingh15/countdown_tasks_3to4-dpo`), sharing the same AdamW and cosine schedule as SFT. RLOO uses the leave-one-out baseline $b_i = \frac{1}{G-1}\sum_{j \neq i} r_j$ with length-normalized importance weights, an entropy coefficient of 0.01, AdamW with constant learning rate 1×10^{-5}, and gradient clipping at 1.0.

GRPO update. The GRPO update worker computes group-relative advantages using population standard deviation (`unbiased=False`) with $\epsilon = 10^{-8}$. Both the current-policy and old-policy log-probabilities are length-normalized over response tokens before computing the importance weight ratio $\rho_i = \exp(\bar{\ell}_\theta - \bar{\ell}_{\text{old}})$; this prevents long completions from dominating the gradient. The clipped surrogate uses $\epsilon_{\text{clip}} = 0.2$. An entropy bonus with coefficient $\alpha = 0.01$ is added to encourage exploration. The KL penalty is disabled ($\beta_{\text{KL}} = 0.0$) by default; the reference model is only instantiated when it is nonzero. The optimizer is AdamW with constant learning rate 1×10^{-5} and weight decay 0.01, gradient clipping at 1.0, batch size 4, and group size $G = 8$. Training runs for 250 gradient steps total. At each step, the update worker logs: policy gradient loss, entropy, KL loss, mean and std of rewards and advantages, mean importance weight, PPO clip fraction, mean current and old policy log-probabilities, gradient norm, and learning rate.

Rollout infrastructure. Rollout generation and gradient updates are handled by two separate Ray remote actors, each assigned one GPU. The `SamplingWorker` loads a vLLM engine (GPU memory utilization 0.9, chunked prefill enabled, max 8192 batched tokens, max 64 sequences) and samples G completions per prompt with temperature 0.6, top-p 0.95, top-k 20, and a hard stop at `</answer>` (stop string included in output). Sequence log-probabilities are extracted from vLLM's per-token logprob output (`logprobs=1`) by summing token-level log-probabilities. The `GRPOUpdateWorker` loads the policy and optional reference model in bfloat16 and holds the AdamW optimizer state. The two workers alternate GPU ownership each step: the active worker is explicitly killed via `ray.kill` before the other is instantiated, keeping peak GPU memory within the H100's capacity.

Curriculum. The training set (`asingh15/countdown_tasks_3to4`) is sorted by the difficulty key (num_operands, target), with stage boundaries computed using `math.ceil` applied to the sorted index list. We use $K = 4$ cumulative stages at fractions $[0.25, 0.50, 0.75, 1.00]$, so stage k trains on the easiest $\lceil f_k N \rceil$ examples and is a strict superset of stage $k-1$. The unlock threshold is $\tau = 0.6$: after every training batch, the batch-mean verifier reward is checked against τ; if exceeded and a later stage exists, the current stage checkpoint is saved and the dataloader is rebuilt from the new larger index set before the next step. The trainer logs `curriculum/stage_index`, `curriculum/stage_fraction`, `curriculum/stage_step`, `curriculum/unlocked`, and `curriculum/stage_complete_reward_mean` (the reward at the moment of unlock) to W&B each step.

Verifier and evaluation. The Countdown verifier extracts the last `<answer>...</answer>` span via regex, validates that the sorted integer literals in the expression exactly match the sorted input numbers, then evaluates the expression using `eval` with an arithmetic character whitelist (`^[\d+\-*/().\s]+$`) and no builtins. Scores are 1.0 (correct answer within 10^{-5} of target), 0.1 (valid format but wrong or unevaluable expression), and 0.0 (missing answer tags). At evaluation time, we load checkpoints into vLLM and sample 16 responses per prompt (temperature 0.6, top-p 0.95, top-k 20) from the held-out test split, saving per-prompt responses and scores as JSON. Pass@k is computed from these saved scores for $k \in \{1, 2, 4, 8, 16\}$.