Extended Abstract

Motivation Verifier-based reinforcement learning offers a scalable way to improve mathematical reasoning without human preference labels: in Countdown, a symbolic checker can automatically verify whether a generated expression uses all numbers exactly once and reaches the target. The main difficulty is reward sparsity. Many prompts yield rollout groups with all-zero rewards, giving weak RLOO updates even with a leave-one-out baseline. We therefore ask whether changing only the training prompt distribution can make verifier-based RLOO more sample-efficient.

Method We study two prompt-distribution interventions while keeping the SFT initialization, verifier, and RLOO objective fixed. First, we apply curriculum sampling over the real Countdown training pool. A symbolic solver labels prompts as easy, medium, or hard, and we compare uniform RLOO, a static curriculum, and an adaptive curriculum. The adaptive controller upweights buckets whose recent success rate is near a target learning frontier, where prompts are more likely to produce mixed correct and incorrect rollouts. Second, we test generated-prompt mixing: Qwen2.5-0.5B-base proposes new Countdown puzzles, which are filtered for solvability and compared against a random solvable-prompt control under the same $50/50$ real/generated training mix.

Implementation For curriculum learning, the solver difficulty labeler enumerates reachable expressions and constructs a structural score from solution counts, minimum complexity, near misses, and arithmetic features such as division, fractions, and negative intermediate values. The full training split is bucketed by quantiles into easy, medium, and hard prompts. The final curriculum runs use 100 RLOO updates, 128 prompts per update, 8 rollouts per prompt, learning rate 10^{-5}, and entropy/KL coefficients 10^{-3}. Evaluation uses 50 held-out real Countdown prompts with 16 responses per prompt.

For prompt generation, the proposer outputs candidate $(\text{nums}, \text{target})$ pairs. A brute-force symbolic solver filters for valid solvable puzzles under in-distribution constraints, producing a pool of 603 unique LLM-generated prompts, matched by a random solvable-prompt pool of the same size. Generated prompts affect training only; evaluation remains on held-out real Countdown prompts.

Results The strongest positive result comes from adaptive curriculum sampling. The vanilla RLOO baseline achieves pass@$1 = 0.4463$, pass@$4 = 0.6680$, and pass@$16 = 0.7400$. The static curriculum improves pass@1 to 0.4850 but reduces pass@4/pass@16 to $0.6484/0.7000$, suggesting narrower coverage. The adaptive curriculum performs best overall, reaching pass@$1 = 0.4913$, pass@$4 = 0.6853$, and pass@$16 = 0.7400$.

Generated-prompt mixing is less reliable. At 15 updates, multi-seed experiments show no statistically reliable improvement over real-only RLOO. At 100 updates, the LLM proposer improves pass@1 over random augmentation (0.430 vs. 0.391), but does not outperform real-only RLOO and trails random augmentation at pass@16. A static curriculum over the small generated pool further hurts coverage.

Discussion These results suggest that prompt selection is more effective than untargeted prompt generation in this setting. Adaptive curriculum works because it reallocates rollout budget within the large and diverse real prompt pool while targeting prompts near the policy's current learning frontier. In contrast, open-loop generated prompts may be valid and solvable but still mismatched to the held-out real distribution or to the policy's current difficulty level. The LLM proposer does encode useful Countdown-like structure, but solvability alone is not enough; useful prompts must also generate informative reward contrast under the current policy.

Conclusion Prompt distribution design is an important but delicate lever for verifier-based RLOO. Adaptive curriculum sampling over real prompts improves pass@1 and pass@4 while preserving pass@16, whereas open-loop LLM-generated prompt mixing does not consistently beat real-only RLOO and can reduce coverage. Future generated-data methods should therefore target generated prompts to the policy's current learnability frontier, rather than simply adding more solvable prompts.

Prompt Distribution Design for RLOO Countdown Reasoning

Zhibo Dai
Department of Energy Science& Engineering
Stanford University
zhibodai@stanford.edu

Yikai Cao
Department of Management Science& Engineering
Stanford University
ykcao@stanford.edu

Abstract

Reinforcement learning with verifiable rewards provides a scalable way to improve language-model reasoning without human preference labels, but sparse binary feedback can make online updates inefficient. We study this issue in Countdown reasoning, where a model must construct an arithmetic expression that reaches a target using each input number exactly once. We compare uniform real-prompt RLOO against two prompt distribution-design strategies: solver-based curriculum sampling over the real Countdown training pool and prompt mixing with solvable puzzles generated by either a random sampler or a Qwen2.5-0.5B proposer.

Our results show that prompt selection is a meaningful design lever. A symbolic solver labels the full training split into easy, medium, and hard buckets, and an adaptive curriculum that prioritizes prompts near the policy's current success frontier improves held-out pass@1 from 0.446 to 0.491 and pass@4 from 0.668 to 0.685. In contrast, open-loop generated-prompt mixing produces valid solvable puzzles but does not consistently outperform real-only RLOO. The LLM proposer improves pass@1 relative to random augmentation at 100 updates, but trails the real-only baseline on broader coverage metrics. Overall, our experiments suggest that verifier-based RLOO benefits more from targeted sampling over a large real prompt pool than from untargeted prompt generation.

1 Introduction

Verifier-based reinforcement learning has become a practical approach for improving language-model reasoning on tasks where correctness can be checked automatically. In such settings, the reward does not require human preference annotation: a symbolic verifier can directly determine whether a generated answer satisfies the task constraints. The Countdown task is a representative example. Given a target number and a set of input numbers, the model must generate an arithmetic expression that uses each number exactly once and evaluates to the target. This makes Countdown a useful testbed for studying reinforcement learning from sparse but reliable correctness signals.

Despite the availability of exact rewards, verifier-based reinforcement learning remains sample-inefficient when most sampled responses are incorrect. In RLOO, the policy samples multiple responses for each prompt and updates using a leave-one-out baseline, so the learning signal depends on within-prompt reward contrast. Prompts that are either too easy or too hard may be uninformative: if all rollouts are correct or all are incorrect, the resulting advantage estimates provide limited

guidance. This suggests that the distribution over training prompts may be as important as the policy-gradient update itself.

In this project, we study prompt distribution design for RLOO fine-tuning on Countdown reasoning. We keep the supervised fine-tuned initialization, symbolic verifier, and RLOO objective fixed, and modify only how prompts are sampled during training. This isolates the effect of the training prompt distribution $q_t(x)$ from other confounding changes in model architecture, reward design, or optimization. We consider two complementary interventions: curriculum sampling over the real Countdown training pool, and generated-prompt mixing using solvable puzzles proposed by either an LLM or a random generator.

Our first intervention uses a symbolic solver to assign difficulty labels to real Countdown prompts. The solver produces structural difficulty features, which are used to bucket prompts into easy, medium, and hard groups. We then compare uniform RLOO with static and adaptive curriculum schedules. The adaptive curriculum estimates recent success rates by difficulty bucket and prioritizes prompts near the policy's current learning frontier. The goal is to allocate rollout budget to prompts that are likely to produce mixed correct and incorrect responses, and hence stronger RLOO advantage signals.

Our second intervention tests whether generated prompts can improve training beyond the real prompt pool. We use Qwen2.5-0.5B-base as an LLM proposer to generate candidate Countdown puzzles, filter them with a symbolic solver for solvability, and compare the resulting pool against a random solvable-prompt control. This experiment separates two possible benefits of generation: the LLM may encode a useful prior over human-like arithmetic puzzles, or the improvement may simply come from adding more diverse solvable prompts. Evaluation is always performed on held-out real Countdown prompts, so generated prompts affect only the training distribution.

Our results show that prompt selection is a useful but delicate lever. Adaptive curriculum sampling over the large real prompt pool improves pass@1 and pass@4 over the uniform RLOO baseline, while preserving pass@16. In contrast, open-loop generated-prompt mixing does not consistently outperform real-only RLOO. Although the LLM proposer generates valid and structured puzzles, its prompts are not targeted to the current policy's difficulty level and can dilute learning on the real distribution. Overall, our findings suggest that verifier-based RLOO benefits more from targeted sampling over a large real prompt pool than from untargeted prompt generation.

2 Related Work

Reinforcement learning for language-model reasoning. Reinforcement learning has been widely used to adapt language models to external feedback, most prominently through RLHF pipelines based on PPO (Schulman et al., 2017; Ouyang et al., 2022). More recent work has shown that simpler REINFORCE-style objectives, including leave-one-out variants, can be competitive for language-model fine-tuning while avoiding some of the complexity of PPO (Williams, 1992; Ahmadian et al., 2024). Our work uses RLOO in this spirit: the policy samples multiple responses per prompt and uses the other responses in the group as a baseline. We do not propose a new policy-gradient estimator. Instead, we ask how the prompts used to collect these rollout groups should be chosen.

Verifier-based mathematical reasoning. Mathematical reasoning tasks are especially suitable for verifier-based training because correctness can often be checked automatically. Prior work has used learned or symbolic verifiers to select or improve mathematical solutions (Cobbe et al., 2021), and recent math-focused language models further demonstrate the value of verifiable feedback for reasoning improvement (Shao et al., 2024). Countdown is a particularly clean instance of this setting: a symbolic checker can determine whether an expression reaches the target while using the given numbers exactly once. However, exact rewards do not remove the main optimization difficulty. Since the reward is sparse, many rollout groups contain no correct answer, making the resulting RLOO update weak. This is the point where prompt selection becomes important.

Curriculum learning and online prompt selection. Curriculum learning studies how the order and difficulty of examples affect training (Bengio et al., 2009), while self-paced learning selects examples according to their estimated easiness during optimization (Kumar et al., 2010). These ideas are closely related to our curriculum extension, but the setting is different. We are not simply reordering a supervised dataset; we are choosing the online prompt stream on which RLOO spends

rollout budget. This distinction matters because a prompt is useful only if it induces informative reward contrast under the current policy. Our adaptive curriculum therefore targets buckets near the policy's current success frontier, rather than following a fixed easy-to-hard schedule.

Generated data and self-improvement. Another line of work improves language models by generating additional training data. STaR bootstraps reasoning from model-generated rationales (Zelikman et al., 2022), Self-Instruct generates instruction-following data with limited human annotation (Wang et al., 2023), and WizardMath uses evolved mathematical instructions and feedback to improve math reasoning (Luo et al., 2023). Our generated-prompt extension is related to this literature, but differs in what is generated and how it is evaluated. We generate new Countdown prompts, not solutions or rationales, and use them only to modify the RLOO training distribution. The held-out evaluation distribution remains the original real Countdown distribution.

Our positioning. Our project is best viewed as a controlled study of prompt distribution design for verifier-based RLOO. We keep the SFT initialization, verifier, reward rule, and RLOO objective fixed, and change only the training distribution $q_t(x)$. This isolates prompt selection from three effects that are often entangled in prior work: the choice of RL optimizer, the design of the verifier, and the construction of new training data. Within this controlled setting, our results suggest a clear distinction. Adaptive selection over a large real prompt pool can improve RLOO by allocating rollouts to prompts near the policy's learning frontier. In contrast, open-loop generated prompts can be valid and solvable without being useful for the current policy or aligned with the held-out real distribution.

3 Method

We isolate the effect of prompt distribution design in verifier-based RLOO. The SFT initialization, symbolic verifier, reward rule, and RLOO objective are kept fixed. At training step t, the only object we modify is the prompt sampling distribution $q_t(x)$. For each sampled prompt x, the policy generates G responses $y_1, \ldots, y_G$, and the verifier assigns rewards $R(y_i, x)$. RLOO then forms the leave-one-out advantage

$$A_i = R(y_i, x) - \frac{1}{G-1} \sum_{j \neq i} R(y_j, x). \tag{1}$$

Thus, a useful prompt is not only one that is solvable, but one that induces informative within-prompt reward contrast under the current policy. We study two ways of changing $q_t(x)$: curriculum selection over the real Countdown training pool, and prompt mixing with generated solvable prompts.

3.1 Curriculum Selection

The curriculum extension changes only the prompt sampler. Instead of sampling uniformly from the real Countdown training split, we first assign each real prompt a difficulty label and then sample from difficulty buckets. The RLOO loss and verifier are unchanged.

Solver-based difficulty estimation. For each Countdown prompt $x = (n_1, \ldots, n_m, T)$, we estimate difficulty using an exact symbolic solver. The solver performs dynamic programming over subsets of the input numbers. For each subset $S \subseteq \{1, \ldots, m\}$, it stores the rational values reachable by arithmetic expressions using exactly the numbers in S. The base case is

$$\mathcal{V}_{\{i\}} = \{n_i\},$$

and larger subsets are constructed by combining two disjoint nonempty subsets S_1, S_2 with $S_1 \cup S_2 = S$:

$$z = z_1 \circ z_2, \qquad z_1 \in \mathcal{V}_{S_1}, \quad z_2 \in \mathcal{V}_{S_2}, \quad \circ \in \{+, -, \times, \div\}.$$

Subtraction and division are evaluated in both orders, and all computations use exact rational arithmetic. This gives a complete symbolic search over expressions that use each input number exactly once.

The difficulty score is then computed from structural features of the solver output. We use three main types of signals: (i) solution abundance, such as the number of solution paths reaching the target; (ii) expression simplicity, such as the minimum depth and minimum complexity among solution paths;

Progress range	$q_t(\text{easy})$	$q_t(\text{medium})$	$q_t(\text{hard})$
$0 \le r_t < 0.4,\ \alpha = r_t/0.4$	$0.70 - 0.35\alpha$	$0.25 + 0.35\alpha$	0.05
$0.4 \le r_t < 0.75,\ \alpha = (r_t - 0.4)/0.35$	$0.35 - 0.25\alpha$	0.60	$0.05 + 0.25\alpha$
$0.75 \le r_t \le 1,\ \alpha = (r_t - 0.75)/0.25$	$0.10 - 0.05\alpha$	$0.60 - 0.20\alpha$	$0.30 + 0.25\alpha$

Table 1: Static curriculum mixture schedule. The schedule moves probability mass from easy and medium prompts toward medium and hard prompts as training progresses.

and (iii) arithmetic difficulty, such as whether all solutions require division, fractional intermediate values, or negative intermediate values. We also include shortcut indicators, such as whether the target can be reached by an addition/subtraction-only expression or by using only two numbers. The final solver difficulty $d(x) \in [0, 1]$ is a clipped weighted score of these normalized features, with larger values corresponding to harder prompts.

The code also supports an SFT-sampling difficulty estimate. Given K responses sampled from the SFT model, we define

$$\widehat{p}(x) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{1}\{R(y_k, x) = 1\}, \qquad d_{\text{SFT}}(x) = 1 - \widehat{p}(x).$$

We use this estimate only for calibration, because running SFT sampling over the full training split would be more expensive than symbolic labeling. On a calibration subset of 2000 prompts, the SFT pass rate decreases monotonically across the solver buckets: 0.458 for easy prompts, 0.369 for medium prompts, and 0.073 for hard prompts. This suggests that the symbolic score is aligned with model-facing difficulty.

Difficulty buckets. After computing $d(x)$ for all labeled training prompts, we divide the real training pool into three quantile buckets:

$$\mathcal{B} = \{\text{easy}, \text{medium}, \text{hard}\}.$$

Let $q_{1/3}$ and $q_{2/3}$ be the empirical one-third and two-third quantiles of the difficulty scores. The bucket assignment is

$$b(x) = \begin{cases} \text{easy}, & d(x) \le q_{1/3}, \\ \text{hard}, & d(x) \ge q_{2/3}, \\ \text{medium}, & \text{otherwise.} \end{cases}$$

Let $\mathcal{D}_b = \{x : b(x) = b\}$ denote the set of training prompts in bucket b.

On the full training split, this procedure labels $490{,}314$ prompts and produces nearly balanced buckets: $165{,}465$ easy, $161{,}410$ medium, and $163{,}439$ hard. We validate the solver-based labels against an independent SFT-sampling estimate on 2000 prompts. The SFT pass rate decreases monotonically across the solver buckets, from 0.458 on easy prompts to 0.369 on medium prompts and 0.073 on hard prompts, suggesting that the symbolic score is aligned with model-facing difficulty.

Static curriculum. The static curriculum is an open-loop schedule. Let T_{train} be the number of training steps and define normalized progress

$$r_t = \min\left\{1, \max\left\{0, \frac{t}{T_{\text{train}} - 1}\right\}\right\}.$$

At each step, each prompt in the batch is sampled by first drawing a bucket according to $q_t(b)$, and then drawing uniformly from $\mathcal{D}_b$. The default schedule used in the implementation is the smooth mixture schedule in Table 1.

Thus, the schedule starts near $(0.70, 0.25, 0.05)$, passes through approximately $(0.35, 0.60, 0.05)$ and $(0.10, 0.60, 0.30)$, and ends near $(0.05, 0.40, 0.55)$. This schedule does not use online feedback from the policy.

Adaptive curriculum. The adaptive curriculum updates the bucket distribution using the policy's recent verifier success rates. For each bucket b, the controller maintains a rolling history of recent

batch-level correctness rates. If bucket b is sampled at step t, its observed correctness rate is

$$\widehat{s}_{b,t}^{\text{new}} = \frac{1}{|\mathcal{G}_t|} \sum_{(x,y)\in\mathcal{G}_t} \mathbf{1}\{R(y,x)=1\}, \tag{2}$$

where $\mathcal{G}_t$ denotes all rollout responses generated from the sampled batch. The rolling estimate $\widehat{s}_{b,t}$ is the average of the most recent such observations for bucket b, using a fixed window.

After an initial warm-up period with uniform bucket sampling, the controller assigns each bucket the score

$$u_{b,t} = -\left|\widehat{s}_{b,t} - s^\star\right|, \tag{3}$$

where $s^\star$ is the target success rate. Buckets whose recent success rates are closest to $s^\star$ are interpreted as being near the policy's learning frontier. The bucket probabilities are then

$$q_t(b) = (1-\epsilon)\frac{\exp(u_{b,t}/\tau)}{\sum_{b'\in\mathcal{B}}\exp(u_{b',t}/\tau)} + \frac{\epsilon}{|\mathcal{B}|}, \tag{4}$$

where τ is the softmax temperature and ϵ is an exploration floor. In the final implementation, the default values are

$$s^\star = 0.40, \qquad \tau = 0.20, \qquad \epsilon = 0.10.$$

The controller samples one bucket $b_t \sim q_t$ per training step and draws the whole prompt batch uniformly from $\mathcal{D}_{b_t}$. After the rollouts are scored, the corresponding bucket history is updated.

The motivation is that very easy prompts often produce all-correct rollout groups, while very hard prompts often produce all-wrong rollout groups. Both cases yield limited leave-one-out advantage signal. By targeting buckets with intermediate recent success, the adaptive curriculum allocates rollout budget to prompts that are more likely to produce mixed correct and incorrect responses.

Prompt-level replay ablation. We also implement a prompt-level replay buffer as an ablation on top of the adaptive curriculum. The replay buffer stores previously seen prompts that appear partially solved. For a prompt x, let $\widehat{p}_t(x)$ be an exponential moving average of its strict pass rate. The replay priority is

$$\text{priority}_t(x) = \widehat{p}_t(x)\big(1-\widehat{p}_t(x)\big). \tag{5}$$

This priority is maximized near $\widehat{p}_t(x) = 0.5$, so replay prefers prompts with high within-prompt reward contrast. When replay is enabled, after the adaptive controller selects a bucket, a fraction of the batch is filled with replayed prompts from that same bucket and the rest is filled with fresh prompts. The RLOO objective remains unchanged; replay only changes which prompts are revisited.

3.2 Prompt Mixing

The prompt-mixing extension changes the training distribution by *augmenting* the real Countdown pool with prompts produced by a separate generator agent. We compare three variants against a real-only RLOO baseline to disentangle whether any gains come from (i) the generator having learned task-relevant structure, (ii) simply having more prompts, or (iii) bolting curriculum selection onto the generated pool.

Proposer. A frozen `Qwen2.5-0.5B-base` model is queried with a few-shot prompt containing example Countdown $(\texttt{nums}, \texttt{target})$ pairs. We sample $n = 128$ continuations per few-shot prompt at $\tau = 1.0$, $p = 0.95$, $k = 40$, scaled to $40 \times 128 = 5{,}120$ attempts in one spike. A tolerant regex parser extracts $(\texttt{nums}, \texttt{target})$; the parser succeeds on 99.6% of generations and 37% of parsed problems are solvable.

Filter. An exact symbolic solver enumerates all permutations $\times$ operator-combinations $\times$ parenthesisations ($\leq 7{,}680$ candidates per problem for $n \leq 4$) and keeps only solvable, in-distribution puzzles ($\texttt{target} \leq 100$, $\max(\texttt{nums}) \leq 99$, matching the $p99$ of the real train split). The result is a fixed pool of 603 unique generated problems.

Random control. To isolate the contribution of the LLM's structural prior, we build a matched control that uniformly samples $(\texttt{nums}, \texttt{target})$ from the same in-distribution range and applies the same solvability filter. Uniform random sampling has a $\sim 10\%$ solvable rate ($\sim 4\times$ lower than the LLM proposer); we draw enough attempts to reach the same 603-item pool size.

Solver wiring (Ablations 2 and 4). The RLOO trainer is unchanged; only `train_dataloader` is patched to a $50/50$ mixture of real and generated prompts, where "generated" is either the LLM-proposer pool (Ablation 4) or the matched random pool (Ablation 2). The test loader stays on the real held-out split so test-side metrics are apples-to-apples against the unaugmented RLOO baseline.

Joint extension (Ablation 5). The joint variant adds curriculum selection on top of the generated pool. We reuse the solver-based structural difficulty scorer from §3.1 on the 603 proposer survivors, quantile-bucketing them into easy/medium/hard at $201/201/201$. We then resample the proposer side with static bucket probabilities $(p_{\text{easy}}, p_{\text{med}}, p_{\text{hard}}) = (0.10, 0.40, 0.50)$—matching the end-state of Zhibo's static curriculum schedule—before applying the same $50/50$ real/generated mix. The schedule is fixed, not adaptive, so the proposer pool is reweighted only once and not re-targeted as the policy moves.

4 Experimental Setup

4.1 Curriculum Training

We evaluate curriculum selection on the Countdown training split using the same SFT initialization and verifier across all curriculum experiments. The policy and reference model are both initialized from `asingh15/qwen-sft-countdown-defaultproj`, with tokenizer `Qwen/Qwen2.5-0.5B`. The training data are drawn from `asingh15/countdown_tasks_3to4`. For the curriculum runs, the only intended difference from the vanilla RLOO baseline is the prompt sampler.

We compare the following curriculum training streams:

- **Uniform RLOO baseline:** prompts are sampled uniformly from the real Countdown training split.
- **Static solver curriculum:** prompts are sampled from easy, medium, and hard buckets using the fixed mixture schedule described before.
- **Adaptive solver curriculum:** bucket probabilities are updated online from recent verifier success rates, targeting buckets near the policy's current learning frontier.
- **Adaptive curriculum with replay:** an ablation that augments adaptive bucket sampling with prompt-level replay of partially solved prompts.

The vanilla baseline is run with the updated original RLOO trainer, while the curriculum runs use extension-local copies of the trainer. This keeps the baseline separate from curriculum-specific sampling changes and leaves the original trainer files unchanged.

Training hyperparameters. All final comparable curriculum runs use the same RLOO training configuration. Each run uses 100 training updates, batch size 128, group size 8, and gradient accumulation over 128 rollout groups. The learning rate is 10^{-5}, with constant learning-rate schedule, weight decay 10^{-4}, no warmup, and no gradient clipping. For rollout sampling during training, we use temperature 1.0, top-$p = 1.0$, and top-$k = -1$. The entropy coefficient and KL coefficient are both set to 10^{-3}.

Adaptive-controller settings. For the adaptive curriculum, we use target success rate $s^\star = 0.40$, temperature $\tau = 0.20$, exploration floor $\epsilon = 0.10$, rolling window size 20, and 10 warmup steps. The replay-buffer ablations use the same adaptive-controller settings. The first replay variant uses replay ratio 0.25, replay warmup 10, and EMA pass-rate eligibility range $[0.05, 0.95]$. The conservative replay variant uses replay ratio 0.10, replay warmup 30, EMA pass-rate eligibility range $[0.20, 0.80]$, and at most 3 replays per prompt.

Evaluation protocol. All models are evaluated on the same held-out real Countdown prompts. Evaluation uses 50 prompts and samples 16 responses per prompt, with temperature 0.6, top-$p = 0.95$, top-$k = 20$, and maximum generation length 1024. We report pass@k, where pass@k is the fraction of prompts for which at least one of the first k sampled responses is functionally correct. Functional correctness is counted strictly by verifier score 1.0. The verifier may assign score 0.1 to responses with the correct answer-tag format but an invalid or wrong equation, so we do not treat `score` > 0 as correct.

4.2 Prompt Mixing

All multi-agent runs warm-start from a locally trained SFT checkpoint on the same `asingh15/countdown_tasks_3to4` train split as the curriculum experiments. The verifier and held-out evaluation protocol are identical to §4.1: 50 held-out prompts, 16 samples per prompt, $\tau = 0.6$, $p = 0.95$, $k = 20$, max 1024 tokens, and pass@k counted with verifier score $= 1.0$.

RLOO hyperparameters. We run the original `rloo_trainer/` with learning rate 5×10^{-6}, $\beta_{\text{KL}} = 5 \times 10^{-3}$, $\beta_{\text{entropy}} = 10^{-3}$, batch size 4, gradient-accumulation 4, group size $K = 16$, and `iw_clip_max` $= 10$ (permissive PPO-style importance-weight clip). All runs are 100 update steps, matching the curriculum experiments.

Conditions. (i) **RLOO baseline** (real prompts only); (ii) **Ablation 2** (real + random-augmentation pool); (iii) **Ablation 4** (real + LLM-proposer pool); (iv) **Ablation 5** (real + curriculum-weighted LLM-proposer pool). Conditions (ii)–(iv) share an identical pipeline aside from the prompt source.

Configuration relative to the curriculum extension. The curriculum and multi-agent extensions use *different* RLOO configurations (curriculum uses larger batches, $K = 8$, lr $= 10^{-5}$, $\beta_{\text{KL}} = 10^{-3}$). Absolute pass@$k$ numbers therefore compare within each extension's baseline, not directly across.

5 Results

5.1 Quantitative Evaluation

All pass@k values are correctness-only counts (verifier score $= 1.0$); format-inclusive (0.1-rewarded) responses are excluded.

Curriculum selection (Extension A). Table 2 reports pass@k across all curriculum conditions trained for 100 updates with the curriculum-extension RLOO configuration.

Table 2: Curriculum-selection results, 100 RLOO updates, batch 128, $K = 8$ rollouts/prompt.

Method	pass@1	pass@4	pass@16
Vanilla RLOO baseline	0.4463	0.6680	0.7400
Static solver curriculum	0.4850	0.6484	0.7000
Adaptive solver curriculum	**0.4913**	**0.6853**	0.7400
Adaptive + replay buffer	0.4600	0.6400	0.7400
Adaptive + conservative replay	0.4800	0.6600	0.7200

The adaptive solver curriculum is the strongest configuration (+4.5 pp pass@1 and +1.7 pp pass@4 over the vanilla baseline). The static curriculum recovers most of the pass@1 gain but loses 4 pp on pass@16, indicating that fixed easy-to-hard shifts narrow the policy's response distribution. Neither replay variant outperforms the non-replay adaptive curriculum: at the 100-step budget, prompt-level replay trades off against fresh prompt coverage and introduces prompt-level sampling bias.

Multi-agent prompt mixing (Extension B). Table 3 reports pass@k for the multi-agent conditions under the multi-agent-extension RLOO configuration.

Table 3: Multi-agent prompt-mixing results, 100 RLOO updates, batch 4×4 grad-accum, $K = 16$, `iw_clip_max` $= 10$, $50/50$ real/generated mixture, single training seed.

Method	pass@1	pass@4	pass@16
SFT (no RL)	0.281	0.585	0.740
RLOO baseline (real only)	**0.431**	**0.685**	**0.780**
+ Random augmentation (Ablation 2)	0.391	0.678	0.760
+ LLM proposer (Ablation 4)	0.430	0.672	0.740
+ LLM proposer + curriculum (Ablation 5)	0.418	0.636	0.680

The pure RLOO baseline is the strongest condition on every k, and is the only one to break the SFT-era pass@16 $= 0.74$ ceiling ($\rightarrow 0.78$). Several observations: (i) the LLM proposer matches the baseline at pass@1 (0.430 vs 0.431) but trails by 4 pp at pass@16—adding generated prompts caps the policy's coverage at the SFT ceiling; (ii) LLM-vs-random *decomposes across* k—LLM beats random at pass@1 ($+3.9$ pp) but loses to random at pass@16 (-2.0 pp), consistent with LLM-shaped cognitively-uniform puzzles tightening single-best responses while narrowing diversity; (iii) static curriculum on the small generated pool (Ablation 5) hurts strictly more, losing 10 pp pass@16 to the baseline—a hard-bucket weight of 0.5 on ~ 201 items causes $\sim 4\times$ over-cycling that sharply narrows the policy's response distribution. The multi-agent ablations report single seeds; from a 15-step multi-seed sweep we measured $\sigma_{\text{pass@1}} \approx 0.02$, so the LLM-vs-baseline gap is not statistically distinguishable and the LLM-vs-random gap is borderline.

5.2 Qualitative Analysis

The generation-side asymmetry between the LLM proposer and the random control was already striking before any training: under the same in-distribution filter, the LLM proposer's puzzles are solvable 37% of the time vs 10% for uniform random sampling—a $\sim 4\times$ gap. Table 4 shows six survivors drawn uniformly at random from each post-filter pool. The LLM consistently emits consecutive-integer sets, small "teaching-example" arities, and human-arithmetic-meaningful targets, even though it was not fine-tuned on this task. The random sampler hits the same range uniformly. This generation-side structural difference does not translate into a held-out pass@k advantage once the prompts are mixed into RLOO (Table 3)—a real implicit prior in the proposer, but the wrong shape of help for what RLOO at this scale needs (which is pass@16 coverage of the held-out distribution, not single-best precision on cognitively-uniform shapes).

Table 4: Six survivors drawn at random from each post-filter generation pool.

LLM proposer	**Random sampler**
`[1, 2, 3, 4]` $\rightarrow 6$	`[73, 7, 49, 12]` $\rightarrow 92$
`[17, 16, 15, 14]` $\rightarrow 11$	`[78, 79, 53, 33]` $\rightarrow 71$
`[46, 47, 48, 49]` $\rightarrow 70$	`[51, 4, 21]` $\rightarrow 26$
`[64, 36, 18, 9]` $\rightarrow 57$	`[14, 9, 51, 93]` $\rightarrow 19$
`[1, 2, 3]` $\rightarrow 7$	`[2, 35, 3, 52]` $\rightarrow 63$

6 Discussion

Both halves of the project target the same lever—the training prompt distribution $q_t(x)$—and arrive at consistent but asymmetric conclusions.

Selection beats generation at this scale. The single strongest result in our experiments is the adaptive solver curriculum ($+4.5$ pp pass@1 over the vanilla RLOO baseline, Table 2). This is a *prompt-selection* intervention: it does not add any new prompts, it only re-weights the $\sim 490{,}000$ real train prompts toward those whose current solver-estimated difficulty matches the policy's learnability frontier. The multi-agent extension instead *generates* new prompts; none of its variants (Ablations 2, 4, 5) significantly improve over their RLOO baseline.

The unknown-distribution problem. The proposer-generated pool comes from a distribution that is opaque relative to the real Countdown distribution beyond the in-distribution filter. Half of every augmented gradient step is therefore fitting an unknown target that the held-out evaluation never queries. At 100 steps the RLOO baseline has converged enough that this dilution becomes a real cost, visible as the -4 pp pass@16 gap between Ablation 4 and the baseline. The random control isolates that the cost is shape-specific rather than purely about volume: random aug loses pass@1 but preserves pass@16, while LLM aug does the opposite.

Why bolt-on curriculum on generated prompts fails. Ablation 5 was designed to test whether the curriculum's frontier-targeting could compensate for the proposer's open-loop mixing. Instead it makes the problem worse. Two limits drive this: the proposer pool only has ~ 201 items per bucket

(vs $\sim 160{,}000$ for the real curriculum), so static 50% weighting on hard over-cycles a small subset; and our scorer is calibrated on SFT pass-rate over the real distribution, so its "hard" label may be uniformly hard (no within-prompt RLOO advantage signal) rather than policy-frontier hard. A useful joint extension would need *both* an adaptive controller (closed-loop bucket re-weighting) *and* a much larger generated pool (thousands of items per bucket, not ~ 200).

Limitations. All multi-agent numbers come from single training seeds under a one-task, small-model, short-horizon (100-step) regime. The two extensions use different RLOO configurations, so absolute pass@k values are not directly comparable across Table 2 and Table 3; within each table the comparisons against that extension's RLOO baseline are matched. The proposer is a frozen Qwen-base model with no fine-tuning loop, leaving substantial headroom for a fully multi-agent self-play variant.

7 Conclusion

In small-batch, verifier-graded RL fine-tuning of a 0.5B model on Countdown, the single empirical lever we found to reliably improve pass@k was *difficulty-aware selection from the real prompt pool* (adaptive solver curriculum, $+4.5$ pp pass@1). Generation-based interventions (an LLM proposer with or without static curriculum weighting) did not exceed the unaugmented RLOO baseline at 100 steps, even though the LLM proposer's generations are $\sim 4\times$ more often solvable than uniform random sampling. The consistent reading across both halves of the project is that, at this scale, the bottleneck is which existing prompts the policy trains on, not how many novel prompts are available—and that targeting new generations at the policy's learnability frontier will need closed-loop control over a much larger generated pool than a single proposer spike provides.

8 Team Contributions

- **Zhibo Dai:** Solver-based difficulty estimation, static and adaptive curriculum controllers, prompt-level replay-buffer ablation, vanilla RLOO baseline (Extension A).
- **Yikai Cao:** LLM proposer + brute-force solvability filter, random-augmentation control, dataset-mixing adapter, RLOO infrastructure (`iw_clip_max` plumbing, Modal deploy/spawn launcher), joint Ablation 5 (curriculum-weighted proposer pool), Extension B writeup.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 12248–12267. doi:10.18653/v1/2024.acl-long.662

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 41–48.

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. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168 [cs.LG]

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems*, Vol. 23.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2023. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. arXiv:2308.09583 [cs.CL]

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,

Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, Vol. 35. 27730–27744.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL]

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 13484–13508. doi:10.18653/v1/2023.acl-long.754

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8, 3–4 (1992), 229–256.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*, Vol. 35.