Extended Abstract

Motivation and problem statement. Language models can produce fluent reasoning traces while still failing exact arithmetic constraints. Countdown is a useful testbed for this problem: the model receives a small set of numbers and a target, then must generate an expression that uses the numbers correctly and evaluates to the target. Because solutions can be checked deterministically, the task fits reinforcement learning with verifiable rewards (RLVR). This project asks whether online reinforcement learning can improve an already supervised-finetuned Countdown model, and whether dynamically choosing problem difficulty can make the reward signal more useful.

Method and novelty. I extend a REINFORCE Leave-One-Out (RLOO) fine-tuning pipeline with an adaptive difficulty curriculum. RLOO samples multiple completions for the same prompt, scores each completion with a verifier, and uses the mean reward of the other completions as a leave-one-out baseline. This produces a group-relative advantage without training a separate value model. My extension adds a sliding-window estimate of recent pass rate. If recent accuracy is below 0.24, the sampler moves toward easier problems; if it is above 0.30, it moves toward harder problems; otherwise the difficulty is kept fixed. The novelty is therefore a lightweight curriculum controller around RLOO, not a new policy-gradient estimator.

Implementation details and headline results. The policy was `asingh15/qwen-sft-countdown-defaultproj`, with `Qwen/Qwen2.5-0.5B` as tokenizer and the same initial policy frozen as the reference model. The dataset was `asingh15/countdown_tasks_3to4`, augmented with a difficulty field and filtered dynamically during training. The implementation added a difficulty-aware dataset wrapper, a curriculum controller, fallback sampling when a difficulty bucket was empty, and reward computation through `compute_score(response,ground_truth)`. As a completed reference result, the IPO/SFT evaluation on 50 held-out prompts achieved pass@1 $= 19/50 = 0.380$, pass@2 $= 27/50 = 0.540$, pass@4 $= 33/50 = 0.660$, pass@8 $= 33/50 = 0.660$, and pass@16 $= 38/50 = 0.760$. The final comparison evaluated Uniform RLOO and Adaptive RLOO at the 128 completed training steps visible in the training curves, each on 50 held-out Countdown prompts with 16 sampled responses per prompt. Uniform RLOO achieved average score 0.256, pass@1 0.18, and pass@16 0.66; Adaptive RLOO achieved average score 0.205, pass@1 0.12, and pass@16 0.60. Thus the controller worked mechanically, but the final adaptive schedule did not outperform uniform sampling.

Discussion, limitations, and conclusion. The implementation shows that adaptive curriculum sampling can be integrated into RLOO without changing the RL objective or adding a value network. The motivation is strongest when rewards are sparse: prompts that are too easy give saturated rewards, while prompts that are too hard give all-zero rewards. The main limitation is that short Modal runs, timeout issues, and limited seed coverage prevented a definitive improvement claim; the report therefore uses held-out prompt confidence intervals rather than seed-level standard deviations. Future work should add longer runs, multiple seeds, fixed-difficulty ablations, staged easy/medium/hard curriculum baselines, and difficulty-specific test splits. Overall, the project suggests that curriculum control is a practical way to make RLVR training more responsive to model progress on verifiable reasoning tasks.

Adaptive Curriculum RLOO for Verifiable Arithmetic Reasoning in Language Models

Lucianna Kelechi Onuoha
Department of Computer Science
Stanford University

Abstract

Arithmetic reasoning remains difficult for language models because fluent natural-language explanations do not guarantee that the final expression satisfies exact symbolic constraints. This project studies the Countdown task, where a model must combine a given set of numbers to reach a target, as a compact testbed for reinforcement learning with verifiable rewards. I extend a REINFORCE Leave-One-Out (RLOO) fine-tuning pipeline with an adaptive difficulty curriculum that changes the training distribution according to recent pass rate. The method uses deterministic rewards from a verifier, group-relative advantages, and a sliding-window curriculum controller. At 128 completed training steps, Uniform RLOO reached average score 0.256, pass@1 0.18, and pass@16 0.66, while Adaptive RLOO reached average score 0.205, pass@1 0.12, and pass@16 0.60 on 50 held-out prompts with 16 completions each. The main significance of the project is that it provides a reproducible implementation of curriculum-controlled RLVR for a small language model, showing how task difficulty can be coupled directly to online learning progress.

1 Introduction

Language models are increasingly used for tasks that require multi-step reasoning, but they can still fail when an answer must satisfy a strict symbolic constraint. A model may produce a convincing chain of thought, yet make a small arithmetic error or use a number incorrectly. This motivates reinforcement learning with verifiable rewards (RLVR), where the reward is computed by a task-specific checker instead of a learned human preference model. In Countdown-style arithmetic tasks, the reward can be exact: an answer is correct if the generated expression uses the provided numbers and evaluates to the target.

This project focuses on improving a supervised-finetuned Countdown model using online reinforcement learning. The baseline model already has some ability to solve the task, so the aim is not to train reasoning from scratch. Instead, the aim is to test whether online sampling and reward feedback can improve reasoning reliability, and whether an adaptive curriculum can make the training signal more useful.

The main research questions are:

1. Can RLOO fine-tuning improve a supervised Countdown model using only deterministic verifier rewards?
2. Does an adaptive difficulty curriculum expose the model to more useful prompts than a fixed training distribution?
3. Where does the method fail qualitatively: invalid formatting, arithmetic mistakes, incorrect number use, or difficulty switching instability?

The hypothesis is that adaptive curriculum sampling should be most helpful when rewards are sparse. If the model is repeatedly trained on prompts it already solves, the reward signal saturates. If it is trained on prompts that are too difficult, most sampled completions receive zero reward and the advantage estimates become uninformative. A curriculum based on recent pass rate is a simple way to keep training near the model's current boundary of competence.

2 Related Work

RLHF and policy optimization for language models. The modern RLHF pipeline popularized by InstructGPT uses supervised fine-tuning, reward modeling, and PPO-style policy optimization to align language models with human preferences (1). While this setup is powerful, it requires preference data and a learned reward model, and PPO adds implementation complexity through value functions, clipping, and KL regularization.

Preference optimization without online RL. Direct Preference Optimization (DPO) showed that preference tuning can be formulated as a simple classification-style objective without explicitly training a reward model or running online RL (2). IPO-style baselines are attractive in small course projects because they are simpler and more stable than full PPO. However, preference methods are offline: they optimize against fixed chosen/rejected examples rather than using new samples from the current policy during training. For verifiable reasoning, online RL is appealing because the model can sample many attempts and receive an automatic reward for each one.

RLVR and group-relative updates. Recent reasoning-focused systems increasingly use verifiable rewards for math and code tasks. DeepSeekMath introduced Group Relative Policy Optimization (GRPO), which removes the value model and uses group-normalized rewards for mathematical reasoning (3). RLOO follows a related motivation: it keeps the online RL loop but forms a leave-one-out baseline from the other completions in the group. This makes it cheaper and easier to implement than PPO, while still allowing the policy to improve from its own sampled attempts (4; 5).

Curriculum learning. Curriculum learning trains models on examples in a structured order rather than a random fixed distribution. In this project, the curriculum is not precomputed from a static schedule. Instead, difficulty is selected online using recent verifier rewards. This positions the project between standard curriculum learning and online RLVR: the task distribution is controlled by the policy's measured progress, not by epoch number alone.

3 Method

3.1 Task formulation

Each example consists of a set of numbers $N = \{n_1, \ldots, n_k\}$ and a target value y. The policy π_θ generates a text response a containing reasoning and a final arithmetic expression. A deterministic verifier computes

$$r(a, y, N) = \begin{cases} 1 & \text{if the final expression is valid and evaluates to } y, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In the implementation, this reward is computed by `compute_score(response,ground_truth)`. This reward is intentionally sparse, because partial reasoning steps are not manually graded.

3.2 RLOO objective

For each prompt x, the model samples a group of G completions $a_1, \ldots, a_G$. Each completion receives reward r_i. The leave-one-out baseline for completion i is the mean reward of the other completions:

$$b_i = \frac{1}{G-1} \sum_{j \neq i} r_j. \tag{2}$$

The advantage is

$$A_i = r_i - b_i. \tag{3}$$

The policy-gradient term increases the likelihood of completions with positive advantage and decreases the likelihood of completions with negative advantage:

$$\mathcal{L}_{\text{RLOO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} A_i \log \pi_\theta(a_i \mid x). \tag{4}$$

The implementation also uses a KL penalty to keep the learned policy close to the frozen reference model:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{RLOO}}(\theta) + \beta_{\text{KL}} D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) - \beta_H H(\pi_\theta), \tag{5}$$

where $\beta_{\text{KL}} = 0.001$ and $\beta_H = 0.001$ in the adaptive run.

3.3 Adaptive difficulty curriculum

The extension adds a controller that tracks recent binary rewards in a sliding window. Let $\hat{p}_t$ be the mean reward over the most recent W samples. The difficulty level is updated using two thresholds:

$$d_{t+1} = \begin{cases} d_t - 1 & \text{if } \hat{p}_t < \tau_{\text{low}}, \\ d_t + 1 & \text{if } \hat{p}_t > \tau_{\text{high}}, \\ d_t & \text{otherwise.} \end{cases} \tag{6}$$

In the final adaptive run, $W = 10$, $\tau_{\text{low}} = 0.24$, and $\tau_{\text{high}} = 0.30$. The dataset was augmented so that each example had a difficulty label; the resulting buckets contained 240,608 easy examples, 249,706 medium examples, and 490,314 hard examples. At each batch construction step, the sampler filters for examples matching the current difficulty. If no examples exist at that difficulty, the implementation falls back to the full dataset so that training does not crash.

Algorithm 1: Adaptive Curriculum RLOO
Input: policy π_θ, reference model π_{ref}, dataset D with difficulty labels, window W, thresholds τ_{low}, τ_{high}
for each training step t:
Estimate recent pass rate $\hat{p}_t$ from the reward window.
Update current difficulty d_t using the threshold rule.
Sample prompts from $D_{d_t} = \{(x, y, d) \in D : d = d_t\}$.
If D_{d_t} is empty, sample from full D.
For each prompt, sample G completions from π_θ.
Compute verifier rewards $r_1, \ldots, r_G$.
Compute leave-one-out advantages $A_i = r_i - \frac{1}{G-1}\sum_{j \neq i} r_j$.
Update θ with RLOO loss, KL penalty, and entropy bonus.
Append rewards to sliding window.
return trained policy.

Figure 1: The adaptive curriculum changes which difficulty bucket is sampled, while the RLOO update itself remains unchanged.

3.4 Novelty relative to prior work

The project does not claim to introduce a new policy-gradient estimator. The novelty is practical: it adds an online curriculum controller to a small-model RLOO pipeline for verifiable arithmetic reasoning. Compared with fixed-difficulty RLVR, the method makes the data distribution depend on recent model performance. Compared with offline preference optimization, it uses new samples from the current policy and immediate verifier rewards. Compared with PPO-style RLHF, it avoids a value network and uses group-relative baselines, making the method easier to run in a limited course-compute setting.

4 Experimental Setup

Dataset and task. Experiments used `asingh15/countdown_tasks_3to4`, a Countdown arithmetic dataset with 3–4 input numbers per example. Each prompt asks the model to combine the provided numbers to reach the target. The dataset was extended with a difficulty field for curriculum sampling.

Models. The policy model was `asingh15/qwen-sft-countdown-defaultproj`, initialized from a supervised-finetuned Countdown model. The tokenizer was `Qwen/Qwen2.5-0.5B`. The reference model for the KL term was the same initial policy model, frozen during RL training.

Baselines. The main comparison is Uniform RLOO versus Adaptive RLOO. Uniform RLOO samples Countdown prompts from the full training distribution, while Adaptive RLOO changes the sampled difficulty bucket according to recent verifier accuracy. I also report the SFT/IPO-style reference pass@k curve because it shows how much correctness existed before this extension. A stronger future ablation would compare adaptive sampling against a fixed staged curriculum with easy examples in the first third of training, medium examples in the second third, and hard examples in the final third.

Metrics. The primary metric is pass@k: a prompt is counted as solved if at least one of k sampled completions is correct. pass@1 measures single-sample reliability, while larger k values measure whether the model can generate a correct solution with repeated attempts. During training, the curriculum controller uses recent mean verifier reward as a local pass-rate estimate.

Training details. The final adaptive RLOO configuration used learning rate 1×10^{-5}, batch size 16, gradient accumulation 4, group size 4, KL coefficient 0.001, entropy coefficient 0.001, temperature 1.0, top-p 1.0, top-$k = -1$, and minimum-$p = 0.0$. Both main runs are reported at 128 completed training steps, matching the point where the available training curves stop. The adaptive run used a sliding window of 10, low threshold 0.24, and high threshold 0.30. The uniform RLOO comparison used the same batch size, group size, and evaluation protocol. A short smoke test also used learning rate 1×10^{-6}, batch size 8, group size 2, and 2 steps to verify that the full training/evaluation pipeline compiled and ran.

5 Results

5.1 Quantitative Evaluation

Table 1 reports the completed baseline pass@k evaluation on 50 held-out prompts. The result shows a large gap between pass@1 and pass@16: the model often has the ability to sample a correct solution, but single-sample reliability is much lower. This pattern supports the motivation for online RL: if correct completions appear among samples, the policy can be updated to make those completions more likely.

Table 1: Completed reference evaluation on 50 held-out Countdown prompts.

Metric	Solved / Total	Accuracy
pass@1	19 / 50	0.380
pass@2	27 / 50	0.540
pass@4	33 / 50	0.660
pass@8	33 / 50	0.660
pass@16	38 / 50	0.760

Table 2: Final evaluation on 50 held-out Countdown prompts with 16 sampled responses per prompt.

Method	Steps	Avg. score	pass@1	pass@2	pass@4	pass@8	pass@16
Uniform RLOO	128	0.256	0.18	0.32	0.46	0.64	0.66
Adaptive RLOO	128	0.205	0.12	0.20	0.28	0.48	0.60

The final evaluation does not support the hypothesis that this adaptive threshold rule improves final accuracy. Uniform RLOO achieved higher average score, pass@1, and pass@16 at the same completed step count. The useful result is therefore diagnostic rather than performance-based: the adaptive controller changed the training distribution, but the chosen thresholds did not translate that control into better held-out performance.

Statistical reliability. Because the final comparison uses one training seed and a 50-prompt evaluation set, I treat these numbers as diagnostic rather than statistically conclusive. The error bars in Figure 4 show Wilson 95% confidence intervals over the held-out prompts. They reflect uncertainty from the finite evaluation set, not standard deviation across independent training runs. I therefore do not claim that the adaptive curriculum is worse in general; instead, the result shows that this particular threshold schedule did not outperform uniform sampling within the completed 128-step run. A stronger evaluation would repeat each method across multiple seeds and report mean $\pm$ standard deviation for pass@k.

Interpreting the pass@k curve. The baseline curve increases from 0.380 at pass@1 to 0.760 at pass@16. This means that the model's search distribution contains many correct solutions, but they are not consistently ranked or sampled first. The flat region from pass@4 to pass@8 suggests that simply sampling more completions does not always help; for some prompts, the model may repeatedly make the same structural error. The adaptive RLOO objective is designed to use the gap between correct and incorrect samples within a group as a learning signal.

Curriculum behavior. The adaptive run used recent reward to control difficulty. When the reward window fell below 0.24, the sampler moved toward easier examples to recover a useful signal. When it exceeded 0.30, the sampler moved toward harder examples to avoid wasting updates on solved prompts. Because the runs are short and single-seed, the curriculum should be interpreted as a functionality result rather than proof that this threshold rule is better. In a longer run, the most important plot would show difficulty over training steps alongside reward/windowed accuracy.

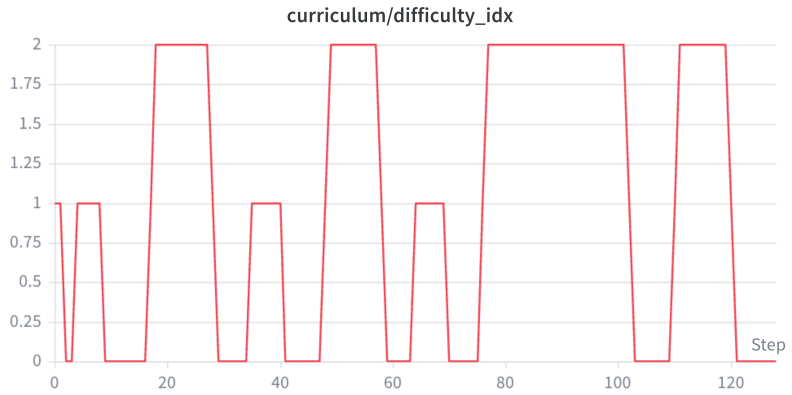

Figure 2: Adaptive curriculum difficulty over the completed training window. The difficulty index moves between easy, medium, and hard buckets, showing that the controller changed the sampled data distribution instead of staying fixed.

Figure 3: Training dynamics from the completed runs. Reward and recent accuracy are noisy rather than monotonically improving; this supports the interpretation that the adaptive controller worked mechanically, but the threshold schedule did not produce a stable performance gain. The original run names in the chart legend reflect launched run names, while the visible traces stop at the 128 completed steps used in the final comparison.

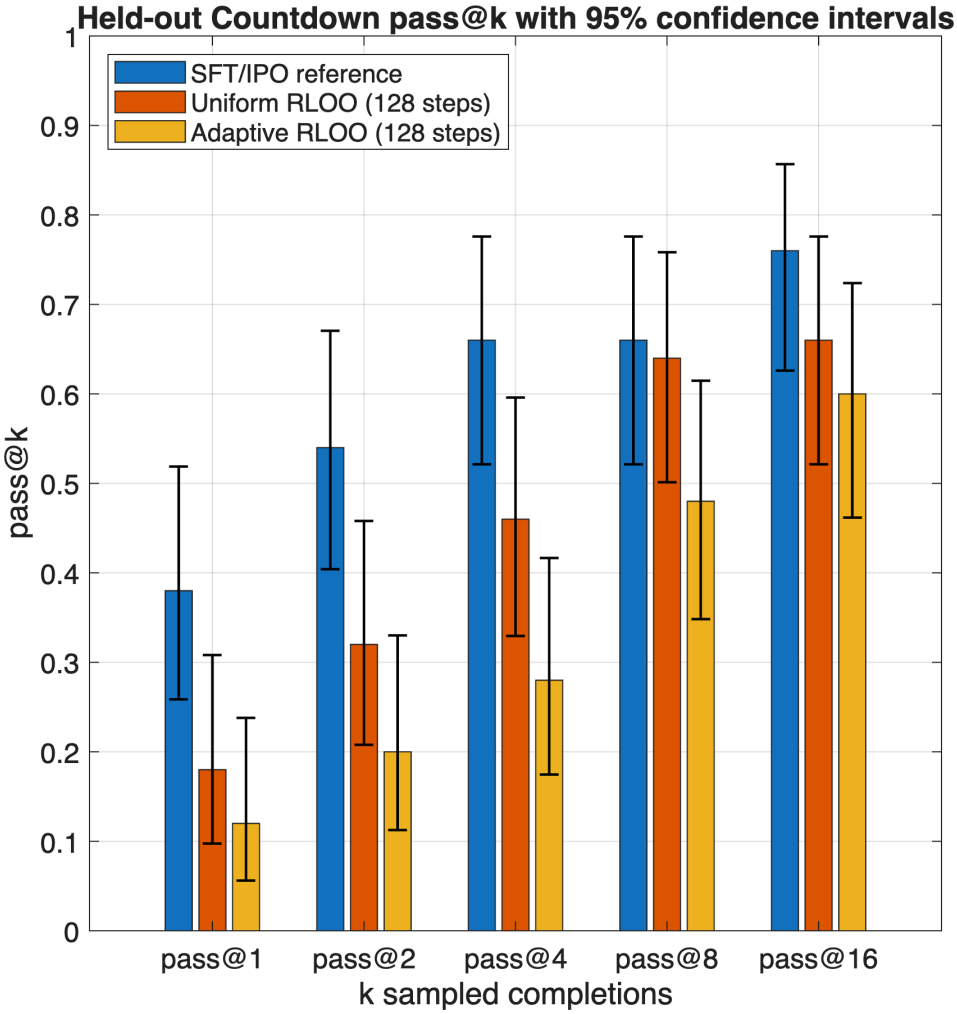

Figure 4: Grouped pass@k comparison with Wilson 95% confidence intervals over 50 held-out prompts. A grouped bar chart is more appropriate than a stacked bar chart because pass@k values are cumulative accuracies rather than additive quantities. The error bars show finite-evaluation uncertainty, not variance across training seeds. The RLOO runs are reported at the 128 completed steps visible in the training curves.

Threshold sensitivity. The adaptive schedule was highly sensitive to the two thresholds. A strict setting such as $\tau_{high} = 0.60$ kept the model on easy prompts because it rarely achieved enough recent successes to move up. A permissive setting such as $\tau_{high} = 0.32$, $\tau_{low} = 0.22$ moved too quickly toward hard prompts and could get stuck with sparse rewards. The best observed mechanical behavior used $\tau_{high} = 0.30$, $\tau_{low} = 0.24$, and window size 10, which produced visible movement across difficulty buckets, although not better final accuracy.

5.2 Qualitative Analysis

Successful rollout. One representative successful completion was:

> **Numbers:** [44, 19, 35], **Target:** 98.
> **Answer:** $(44 + 19) + 35 = 98$.
> **Reward:** 1.0.

This example is simple, but it shows the ideal structure for verifier-based RL: the response uses the allowed numbers exactly once, produces a valid expression, and reaches the target. For such examples, the reward signal is unambiguous.

Table 3: Compact qualitative examples from Countdown rollouts.

Type	Prompt	Model behavior	Reward
Correct solution	$[44, 19, 35] \rightarrow 98$	Outputs $(44 + 19) + 35$, uses each number once, and evaluates exactly to 98.	1.0
Arithmetic error	$[63, 95, 96] \rightarrow 64$	Produces plausible arithmetic but verifies an expression that does not evaluate to the target.	0.1
Invalid or repeated number use	Countdown prompt	Reaches a nearby or correct-looking value by introducing constants or reusing numbers outside the allowed set.	0.1

Failure modes. The main failure cases observed in this task fall into four categories:

1. **Arithmetic error:** the model writes a plausible expression that does not evaluate to the target.
2. **Invalid number use:** the model reaches the target but repeats a number or uses a number not provided in the prompt.
3. **Formatting error:** the model provides reasoning but the verifier cannot reliably extract a final expression.
4. **Difficulty mismatch:** the sampler gives prompts that are either already too easy or too hard, causing saturated rewards or all-zero rewards.

The adaptive curriculum directly targets the fourth failure mode. It does not solve verifier brittleness or expression-formatting errors by itself, so these should be handled with stricter answer templates or more robust parsing.

Why the method can work. RLOO is well-suited to Countdown because multiple completions for the same prompt can differ sharply in reward. If one response solves the problem and the other responses fail, the successful response receives a positive advantage. Unlike a global batch baseline, the leave-one-out baseline is prompt-local, so it compares completions that attempted the same problem. This is important because a hard problem and an easy problem should not be compared as if their raw rewards were equally informative.

Why the method can fail. The method can fail when all completions in a group receive the same reward. If all rewards are zero, every completion looks equally bad and the relative learning signal is weak. If all rewards are one, every completion looks equally good and the policy receives little information about which reasoning style is better. This is exactly why the curriculum controller is useful: it tries to keep prompts in a regime where the group contains both successes and failures.

6 Discussion

This project was limited mainly by compute and infrastructure. Modal timeouts and file-retrieval issues made it difficult to run long experiments, multiple seeds, and complete ablations. The final comparison is short-horizon and single-seed: both curves are reported at the 128 completed steps visible in the logs. As a result, the current evidence is strongest for implementation correctness and qualitative plausibility, not for a statistically reliable improvement claim. In particular, the absence of multiple seeds means that the observed gap between Uniform RLOO and Adaptive RLOO should not be interpreted as a definitive ranking of the methods. A second limitation is that the reward is binary. Binary rewards make evaluation clean, but they do not distinguish between a response that is one arithmetic step away and a response that is completely malformed. A third limitation is that difficulty labels may not perfectly reflect model-specific difficulty; a prompt that is mathematically simple for a human can still be hard for a language model because of formatting or tokenization. One concrete improvement would be to add a staged curriculum baseline that uses easy prompts in the first third of training, medium prompts in the second third, and hard prompts in the final third. This would clarify whether the adaptive controller adds value beyond a simple hand-written easy-to-hard schedule.

The broader impact of this work is mostly methodological. Verifiable-reward training can improve reasoning without human labels, which makes it attractive for math, code, and symbolic tasks. However, optimizing against a verifier can also encourage reward hacking if the checker is incomplete. For future versions, the verifier should explicitly check number use, expression validity, and final-answer formatting.

7 Conclusion

This project implemented an adaptive curriculum extension for RLOO fine-tuning on Countdown arithmetic reasoning. The baseline pass@k results show that the model can often sample correct solutions, but does not reliably produce them at pass@1. Adaptive RLOO is a natural response to this gap: it uses online verifier rewards to reinforce successful completions and adjusts task difficulty using recent performance. The main take-home message is that curriculum control can make RLVR training more responsive and interpretable without adding a value model or changing the core RLOO objective. Future work should run longer training, multiple seeds, difficulty-specific test sets, and parser robustness experiments. The most important ablation is a fixed staged curriculum baseline: train on easy problems for the first third, medium problems for the second third, and hard problems for the final third. That would test whether adaptivity itself helps beyond simply ordering the data from easy to hard.

8 Team Contributions

Lucianna Kelechi Onuoha: completed the full project implementation and report, including the baseline evaluation, RLOO training setup, adaptive curriculum design, dataset difficulty filtering, reward integration, Modal/W&B debugging, qualitative analysis, and final write-up.

Changes from Proposal. The proposal focused on comparing standard fine-tuning and RL-style improvements for Countdown reasoning. The final project narrowed the extension to adaptive curriculum RLOO because it was more feasible within the compute budget and better matched the verifiable reward structure of the task. The main change was therefore a shift from broad method comparison to a deeper implementation of one online RL extension.

A Reproducibility Notes

The key implementation files were `rloo.py`, `rloo_dataset.py`, `rloo_update_worker.py`, `sampling_worker.py`, and `ipo.py`. The project used Modal for remote training and W&B for logging. Training outputs were saved to Modal volumes, including `default-proj-training` and the RLOO checkpoint directory `/vol/checkpoints/rloo_checkpoints`. The most important implementation checks were: ensuring `numpy` was imported as `np`, ensuring `compute_score`

was available in the worker, checking that `all_sample_log_probs` contained numeric values, and compiling the modified Python files before launching remote jobs.

B Additional Figure Notes

The main report includes the grouped pass@k comparison with Wilson confidence intervals, the adaptive difficulty trace, the training reward/recent-accuracy diagnostics, and a compact qualitative table. The confidence intervals are over held-out evaluation prompts; they are not a substitute for multiple random seeds, but they directly address uncertainty from the finite 50-prompt test set.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*.

[2] 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*.

[3] 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 preprint arXiv:2402.03300*.

[4] 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. *arXiv preprint arXiv:2402.14740*.

[5] Hugging Face TRL. 2024. RLOO Trainer Documentation. `https://huggingface.co/docs/trl/en/rloo_trainer`.