Extended Abstract

Motivation Large language models remain unreliable on tasks requiring exact multi-step arithmetic, planning, and symbolic manipulation. The Countdown task provides a controlled setting for studying these limitations: given a target value and a set of input numbers, a model need to generate an arithmetic expression that reaches the target while satisfying strict number-use constraints. Because generated expressions can be parsed and checked automatically, Countdown enables verifier-based reinforcement learning with objective rewards for functional correctness. However, these rewards are often sparse, especially early in training, when difficult prompts may yield only incorrect rollouts. The default pipeline compares supervised fine-tuning (SFT), preference optimization (IPO), and verifier-based online RL with REINFORCE Leave-One-Out (RLOO). Our extension asks whether curriculum learning and self-improvement can improve RLOO under sparse symbolic rewards.

Method We evaluate five training strategies: IPO, baseline RLOO, Heuristic Curriculum RLOO, Self-Improvement RLOO, and Phased Adaptive Self-Improvement RLOO initialized from SFT. IPO learns from chosen–rejected response pairs. RLOO samples multiple completions per prompt and estimates relative advantages using verifier rewards with a leave-one-out baseline. Heuristic Curriculum RLOO orders examples by available difficulty metadata, or otherwise by a hand-designed arithmetic difficulty score, and progressively unlocks cumulative easy-to-hard stages. Self-Improvement RLOO augments real prompts with generated solvable stepping-stone tasks to densify rewards. Phased Adaptive Self-Improvement further adds a warm-up phase and soft filtering, enabling early exploration while later updates prioritize high-signal mixed groups.

Implementation All methods are initialized from the same SFT-tuned Qwen2.5-0.5B backbone and fine-tuned on Countdown-style data. IPO is trained on a separate preference dataset of chosen–rejected response pairs. For RLOO-based methods, vLLM samples multiple responses per prompt, a rule-based Countdown verifier assigns rewards, and the Hugging Face policy is updated with leave-one-out advantage estimates, entropy regularization, and a KL penalty. The self-improvement extension uses a deterministic local teacher to generate guaranteed-solvable synthetic arithmetic tasks. These synthetic tasks are mixed with real prompts early in training and gradually annealed as optimization progresses.

Results On a fixed local evaluation set of 50 prompts with 16 samples per prompt, IPO achieved 36.25% rollout accuracy. RLOO improved rollout accuracy to 48.25%. Heuristic Curriculum RLOO underperformed, reaching only 29.88% rollout accuracy. Self-Improvement RLOO achieved 55.25% rollout accuracy and the highest prompt-level pass@16, at 80.0%. Phased Adaptive Self-Improvement initialized from SFT provided the strongest single-sample reliability, achieving 59.25% rollout accuracy, but had a lower pass@16 of 72.0%.

Discussion These experiments suggest that curriculum design matters more than simply imposing an easy-to-hard ordering. A hand-designed difficulty score can expose the model to simpler prompts early in training, but it does not necessarily provide a useful RL signal. As stages broaden, the policy may still encounter sparse rewards and unstable updates. In contrast, self-improvement alters the training distribution by adding solvable stepping-stone problems, thereby increasing the frequency of rewarded rollouts during the early stages of optimization. The phased adaptive filtering results further indicate that high-signal mixed groups are beneficial only after sufficient warmup. Applying aggressive filtering too early can deprive optimization of adequate learning signal.

Conclusion Among the evaluated extensions, Self-Improvement RLOO performs best in terms of pass@16, whereas Phased Adaptive Self-Improvement initialized from SFT achieves the highest single-sample rollout accuracy. Importantly, both methods outperform the baseline approaches on their respective primary metrics, showing that adaptive or self-improving curricula provide stronger optimization signals than standard IPO or RLOO alone. The main negative result is also informative: fixed heuristic difficulty ordering underperforms even IPO in rollout accuracy. This finding motivates curricula that either generate learnable intermediate tasks or adapt to the current policy's reward patterns.

Verifier-Based Reinforcement Learning for Countdown with Curriculum Training

Yuyan Wu
Department of Civil and Environmental Engineering
Stanford University
wuyuyan@stanford.edu

Abstract

In this project, we aim to improve the arithmetic reasoning ability of a language model on the Countdown task, where the model needs to construct a valid expression reaching a target number from given inputs. Countdown is a useful reasoning benchmark because it requires multi-step arithmetic planning. However, verifier-based RL is difficult due to sparse rewards from mostly incorrect early rollouts. Starting from the default pipeline, we implement SFT, IPO, and RLOO, then study curriculum-learning extensions for RLOO, including fixed heuristic difficulty ordering, Self-Improvement RLOO with generated solvable stepping-stone prompts, and Phased Adaptive Self-Improvement RLOO with warm-up and mixed-group filtering. On a fixed 50-prompt evaluation with 16 samples per prompt, RLOO improves over IPO from 36.25% to 48.25% rollout accuracy, while Heuristic Curriculum RLOO drops to 29.88%. Self-Improvement RLOO reaches 55.25% rollout accuracy and the best pass@16 of 80.0%, whereas Phased Adaptive Self-Improvement achieves the best rollout accuracy of 59.25% but lower pass@16. These results suggest that curriculum learning helps when it increases the density of learnable rewarded trajectories, rather than simply enforcing a static easy-to-hard order.

1 Introduction

Reinforcement learning has become an important post-training technique for improving language models beyond supervised imitation, especially on tasks where correctness cannot be fully captured by next-token prediction alone. By optimizing models with preference feedback or verifiable rewards, RL can encourage outputs that better satisfy task-specific objectives such as reasoning correctness, instruction following, and final-answer validity. In this project, we study verifier-based RL fine-tuning for mathematical reasoning on the Countdown arithmetic task. In Countdown, each prompt provides a target value and a set of input numbers, and the model must generate a valid arithmetic expression that uses the given numbers exactly once and evaluates to the target. For example, given numbers $[63, 95, 96]$ and target 64, a valid solution is $63 + (96 - 95) = 64$. Countdown is a representative benchmark for LLM reasoning because it requires the model to perform multi-step arithmetic planning, satisfy strict symbolic constraints, and generate an executable expression rather than simply recall factual knowledge.

The objective of this project is to extend verifier-based reinforcement learning for language-model mathematical reasoning through curriculum learning during RL training. We build on the default pipeline that fine-tunes Qwen 2.5 0.5B model with supervised fine-tuning, preference optimization, and online reinforcement learning. First, supervised fine-tuning warm-starts the model on correct Countdown completions. Next, IPO trains the model from pairwise preference data by increasing the relative likelihood of preferred responses over rejected ones. Finally, RLOO samples multiple candidate solutions online and applies a policy-gradient update using a leave-one-out baseline within each response group. RLOO can directly leverage rule-based verifier rewards to reinforce correct

solutions, but its effectiveness depends on receiving informative reward variation among sampled rollouts. This signal is highly sensitive to prompt difficulty: easy prompts may become uninformative once the model solves them reliably, while hard prompts often produce mostly zero-reward rollouts early in training. In both cases, the resulting advantage estimates can be weak or unstable, limiting the efficiency of RL fine-tuning.

To address this sparse-reward challenge, our extension investigates whether curriculum learning can improve the efficiency and stability of verifier-based RL fine-tuning. The central idea is to train RLOO on a difficulty-aware sequence of Countdown prompts so that the model first encounters problems that are more likely to produce informative reward variation, while eventually training on the full range of task difficulty. We evaluate several curriculum variants, including static heuristic ordering, synthetic self-improvement, and phased adaptive filtering. Our results show that curriculum design is critical: while standard RLOO improves rollout accuracy over IPO from 36.25% to 48.25%, a simple heuristic curriculum drops to 29.88%, suggesting that static easy-to-hard ordering can hurt performance. In contrast, Self-Improvement RLOO and Phased Adaptive training further improve rollout accuracy to 55.25% and 59.25%, respectively. These findings suggest that effective curricula should expose the model to prompts that are challenging enough to provide a useful learning signal while still remaining solvable during training.

2 Related Work

The default pipeline reflects two broad post-training paradigms for language models. Preference optimization methods such as DPO and IPO learn from relative feedback without training a separate reward model, while online RL methods such as PPO and REINFORCE-style optimization sample from the current policy and directly optimize a reward signal (Rafailov et al., 2023; Gheshlaghi Azar et al., 2023; Ahmadian et al., 2024). RLOO is particularly relevant in this setting because it uses multiple responses to the same prompt and a leave-one-out baseline to reduce the variance of policy gradient estimates (Ahmadian et al., 2024). Our work studies this online RL setting with verifiable rewards, where correctness can be computed automatically.

Countdown-style arithmetic reasoning provides a compact testbed for studying search, self-verification, and reinforcement learning in language models (Gandhi et al., 2024; Jiayi Pan, 2025). The task is especially suitable for verifier-based RL because each generated expression can be checked exactly against the target value, avoiding the need for learned reward models or human preference labels. Small-scale RLVR systems such as TinyZero show that rule-based rewards can induce measurable reasoning and self-verification behavior when the reward is well aligned with the evaluation metric (Jiayi Pan, 2025).

However, verifier-based RL also exposes a central limitation of outcome-only feedback: the reward is often sparse and binary. Prior work on process and outcome supervision for mathematical reasoning shows that final-answer feedback provides limited information about which intermediate reasoning steps caused success or failure (Uesato et al., 2022; Lightman et al., 2023). Recent work on exploration for LLM reasoning further argues that sparse outcome-based rewards and limited exploration can push policies toward repetitive or suboptimal reasoning patterns (Zhang et al., 2025). In Countdown, this problem appears when difficult prompts produce mostly all-zero rollout groups, leaving RLOO with little useful advantage signal.

Curriculum learning offers a natural way to address this issue by changing the training distribution over time. The basic idea is to order or select examples according to learnability, often starting from easier examples and gradually moving toward harder ones (Bengio et al., 2009). In reinforcement learning, curricula have been used to reduce exploration difficulty, stabilize optimization, and mitigate sparse-reward failure modes (Narvekar et al., 2020; Portelas et al., 2020). Curriculum design can take the form of manually staged task progressions, teacher-student task selection, reverse curricula from goal states, or self-play mechanisms that generate increasingly challenging tasks (Florensa et al., 2017; Matiisen et al., 2019; Sukhbaatar et al., 2017).

Recent work has adapted these ideas to LLM reasoning and RL fine-tuning. Online difficulty filtering and AdaRFT argue that RL training is most effective on examples that are challenging but still solvable, rather than examples that are either already trivial or consistently unsolved (Bae et al., 2026; Shi et al., 2025). Self-Evolving Curriculum learns an online curriculum by formulating category selection as a non-stationary multi-armed bandit problem and using advantage magnitude as a proxy

for learning gain (Chen et al., 2025). DUMP extends this idea to distribution- level scheduling with expected absolute advantage and UCB-based exploration (Wang et al., 2025), while SOAR uses a teacher-student meta-RL framework to generate synthetic stepping-stone problems grounded in student improvement on hard target tasks (Sundaram et al., 2026). These works suggest that curriculum design can improve verifier-based RL, but many require learned teachers, online bandit selection, or complex generation pipelines. In contrast, our extension studies a simpler and more controlled question: whether difficulty-aware sampling and self-improvement with synthetic solvable prompts can improve RLOO on Countdown under the same model, verifier, and training budget as the uniform-sampling baseline.

3 Methods

To study RL fine-tuning on Countdown, we first establish the default post-training pipeline with three baselines: supervised fine-tuning (SFT), implicit preference optimization (IPO), and verifier-based online reinforcement learning with REINFORCE Leave-One-Out (RLOO). SFT provides both a baseline policy and the initialization/reference model for subsequent post-training. IPO then learns from pairwise preference data, while RLOO directly optimizes rule-based verifier rewards through online sampling. The curriculum and self-improvement methods introduced below are all built on the RLOO stage: they keep the same model backbone, verifier, and leave-one-out update, but change how prompts are sampled or emphasized during RL training.

3.1 Task and Verifier Reward

We use the default Countdown 3-to-4 number datasets provided in the project pipeline. Each example is represented as a structured arithmetic prompt containing a target value and a list of allowed numbers. Although all training stages use the same underlying Countdown problem format, the data are organized differently for each method. The SFT stage uses expert completions for supervised warm-starting. The IPO stage uses pairwise preference examples consisting of a prompt, a preferred response, and a rejected response. The RLOO-based methods use only prompts during training: the current policy samples candidate solutions online, and each sampled response is scored by the rule-based verifier.

Generated responses are required to place the final expression inside `<answer>...</answer>`. The verifier extracts this answer span and checks whether the expression uses exactly the allowed numbers, is evaluable, and equals the target. The RLOO reward is three-level: responses without an answer tag receive 0; responses with an answer tag but invalid number usage, unparsable arithmetic, or an incorrect value receive the default format reward of 0.1; and fully correct responses receive 1.0. For rollout accuracy and group filtering, we binarize rewards by treating only responses with score 1.0 as correct.

3.2 Baselines: SFT, IPO, and RLOO

We compare our curriculum-based methods with three post-training baselines: supervised fine-tuning (SFT), implicit preference optimization (IPO), and REINFORCE Leave-One-Out (RLOO). All baselines use the same Countdown prompt and response format but differ in their supervision signal. SFT learns from expert demonstrations, IPO learns from offline pairwise preferences, and RLOO learns from online rollouts scored by the rule-based Countdown verifier.

Supervised Fine-Tuning. SFT adapts the base Qwen2.5-0.5B model to the Countdown response format and arithmetic reasoning style. Given a prompt x and an expert completion $y = (y_1, \ldots, y_{|y|})$, we minimize the completion-only negative log-likelihood:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathrm{SFT}}}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid x, y_{<t})\right].$$

Prompt tokens are masked out of the loss, so the model is trained only to predict the completion conditioned on the prompt. The resulting SFT checkpoint is used to initialize IPO and RLOO and serves as the frozen reference policy whenever a reference model is required.

Implicit Preference Optimization. IPO is an offline preference-optimization baseline trained on triples (x, y_w, y_l), where y_w is preferred over y_l. For a completion y, we define its sequence log-probability under policy π_θ as

$$\log \pi_\theta(y \mid x) = \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}).$$

The policy-to-reference log-ratio is

$$r_\theta(x, y) = \log \pi_\theta(y \mid x) - \log \pi_{\text{ref}}(y \mid x).$$

The preference margin is then

$$m_\theta(x, y_w, y_l) = r_\theta(x, y_w) - r_\theta(x, y_l).$$

IPO minimizes the squared deviation between this margin and the target margin:

$$\mathcal{L}_{\text{IPO}}(\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{IPO}}} \left[\left(m_\theta(x, y_w, y_l) - \frac{1}{2\beta} \right)^2 \right],$$

where β controls the strength of the preference constraint. This objective increases the relative probability of preferred completions over rejected completions while anchoring the policy to the SFT reference. Unlike RLOO, IPO does not sample new responses during training and does not directly optimize the Countdown verifier reward.

REINFORCE Leave-One-Out. RLOO is an online RL baseline that uses the rule-based Countdown verifier as the reward function. For each prompt x, a rollout policy μ generates a group of K completions $\{y_1, \ldots, y_K\}$. Each completion is scored by the verifier, yielding rewards $\{R_1, \ldots, R_K\}$. For completion i, the leave-one-out baseline is the mean reward of the other completions for the same prompt:

$$b_i = \frac{1}{K-1} \sum_{j \neq i} R_j,$$

and the corresponding advantage is

$$A_i = R_i - b_i.$$

Equivalently, if

$$\bar{R} = \frac{1}{K} \sum_{j=1}^{K} R_j,$$

then

$$A_i = \frac{K}{K-1}(R_i - \bar{R}).$$

Thus, A_i measures whether completion i performs better or worse than the other samples for the same prompt.

When rollouts are generated on-policy, no importance correction is needed. If rollouts are generated by a stale or otherwise distinct behavior policy μ, we can use a sequence-level importance weight:

$$\rho_i(\theta) = \exp\left(\log \pi_\theta(y_i \mid x) - \log \mu(y_i \mid x)\right).$$

The corresponding policy-gradient estimator is

$$\widehat{\nabla_\theta J} = \frac{1}{K} \sum_{i=1}^{K} \rho_i(\theta) A_i \nabla_\theta \log \pi_\theta(y_i \mid x),$$

with $\rho_i(\theta) = 1$ in the strictly on-policy case. In practice, the update may include an entropy bonus to encourage exploration and a KL penalty against the SFT reference policy to limit policy drift.

RLOO is well-suited to Countdown because each sampled expression can be checked automatically. Its useful learning signal. However, it depends on reward variation within each group. If all completions in a group receive the same reward, then all leave-one-out advantages are zero, and the group contributes no policy-gradient signal apart from regularization. Mixed groups, in which some completions are correct and others are incorrect, provide the clearest RLOO learning signal. This observation motivates the curriculum and self-improvement variants introduced below.

3.3 Heuristic Difficulty Curriculum

Verifier-based RLOO can suffer from sparse rewards early in training: for difficult prompts, all sampled rollouts are often incorrect, producing all-zero rewards and little useful relative-advantage signal. To make early updates more informative, we implement a heuristic easy-to-hard curriculum. The trainer assigns each training prompt a scalar difficulty score, sorts the dataset by this score, and trains on cumulative subsets of increasing difficulty.

We estimate difficulty using the proxy

$$d(x) = 1000 \cdot |\text{numbers}| + |\text{target}| + 0.01 \cdot \max_i |\text{number}_i|.$$

This heuristic prioritizes prompts with fewer numbers and smaller target magnitudes, while using the largest input number only as a small tie-breaking term. With three curriculum stages, stage 0 samples from the easiest third of the dataset, stage 1 samples from the easiest two thirds, and stage 2 samples from the full dataset. The motivation is that easier prompts are more likely to produce mixed reward groups early in training, giving RLOO a clearer learning signal than uniformly sampling from the full dataset. However, because the difficulty score is static and hand-designed, it may not accurately reflect the model's actual competence during training.

3.4 Self-Improvement with Synthetic Solvable Prompts

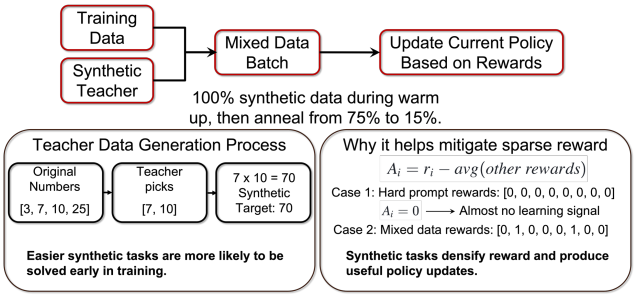

Figure 1: Self-Improvement RLOO mitigates sparse rewards by mixing real Countdown prompts with easier teacher-generated synthetic prompts. The synthetic ratio is high during warmup and gradually annealed, increasing the chance of mixed-reward rollout groups and providing more informative RLOO advantage signals early in training.

Self-Improvement RLOO modifies the RLOO training distribution by adding synthetic prompts generated by a deterministic local teacher. As illustrated in Figure 1, each training batch is formed by mixing real Countdown examples with synthetic teacher-generated examples, and the current policy is then updated using the same verifier-based RLOO objective. The goal is not to replace the original task, but to provide easier stepping-stone prompts early in training, when sparse rewards make standard RLOO updates unstable or uninformative.

The synthetic teacher constructs prompts by sampling a subset of numbers and combining them with arithmetic operations to produce a known valid expression and target. For example, given the original numbers $\{3, 7, 10, 25\}$, the teacher may select $\{7, 10\}$ and construct the expression $7 \times 10 = 70$, yielding a synthetic Countdown prompt with target 70. These prompts are typically easier than the full training examples because they can use fewer numbers during early training. Over time, the synthetic difficulty is increased by allowing more numbers in the generated prompts, making the auxiliary tasks closer to the original Countdown distribution.

The mixing schedule is designed to give the model dense learning signal early while gradually returning emphasis to the real training distribution. During the initial warmup phase, batches can be fully synthetic, which exposes the policy to prompts that are more likely to receive nonzero verifier rewards. After warmup, the synthetic ratio is annealed, for example from 75% to 15%, so that real Countdown prompts make up an increasing fraction of the training data. This schedule encourages early exploration and reward discovery without permanently shifting the optimization objective away from the original task.

This design is motivated by the structure of RLOO advantages. For each prompt, RLOO samples multiple rollouts and assigns the advantage

$$A_i = r_i - \frac{1}{K-1}\sum_{j \neq i} r_j,$$

where r_i is the reward of rollout i and K is the number of sampled responses for the same prompt. If all rollouts for a hard prompt are incorrect, the reward group becomes $[0, 0, \ldots, 0]$, and all leave-one-out advantages are zero. Such all-zero groups provide almost no useful signal about which responses should be encouraged or discouraged. In contrast, an easier synthetic prompt is more likely to produce a mixed group, such as $[0, 1, 0, \ldots, 0]$. In this case, the correct rollout receives a positive relative advantage while incorrect rollouts receive negative relative advantages, producing a clearer policy-gradient update. Thus, synthetic solvable prompts densify the reward signal and increase the frequency of informative mixed-reward groups during the early stages of RLOO training.

3.5 Phased Adaptive Mixed-Group Filtering

Phased Adaptive Mixed-Group Filtering keeps the synthetic-prompt self-improvement setup, but changes which sampled prompt groups are used for the RLOO update. After sampling multiple rollouts for each prompt, the trainer classifies the resulting reward group as all-zero, all-one, or mixed. Mixed groups are prioritized because they contain both successful and unsuccessful responses under the same prompt, giving RLOO the clearest relative-advantage signal.

Directly filtering for only mixed groups can be unstable early in training, since an SFT-initialized policy may produce too few such groups. Therefore, the method uses a phased schedule. During warmup, no filtering is applied, allowing the policy to learn from the full mixture of real and synthetic prompts. After warmup, the trainer preferentially keeps mixed groups, retains a small fraction of all-correct real-prompt groups, and skips most all-correct synthetic groups, which are likely to be too easy. If the filtered update set becomes too small, additional real-prompt groups are added as fallback examples to maintain a stable effective batch size.

The method also records all-zero real-prompt groups in a hard-prompt buffer for diagnostic tracking and potential resampling. Overall, this phased filtering strategy aims to concentrate updates on high-signal prompt groups while avoiding optimization starvation or overfitting to easy synthetic tasks.

4 Experimental Setup

4.1 Dataset and Model Setup

All experiments are conducted on the Countdown 3-to-4 number arithmetic task using Qwen2.5-0.5B as the base model. SFT is trained on `Asap7772/cog_behav_all_strategies`, IPO uses the paired preference dataset `asingh15/countdown_tasks_3to4-dpo`, and all RLOO-based methods train on `asingh15/countdown_tasks_3to4`. IPO and all RLOO variants are initialized from the SFT checkpoint `asingh15/qwen-sft-countdown-defaultproj`. All reported checkpoints are evaluated with the same fixed local evaluation script. The evaluation set contains 50 held-out prompts, and each method samples 16 responses per prompt, giving 800 sampled completions per method.

Generations are scored by a rule-based Countdown verifier. The verifier extracts the final `<answer>...</answer>` span, checks that the generated expression uses exactly the provided numbers, safely evaluates the arithmetic expression, and compares the result to the target with tolerance 10^{-5}. The reward is 0.0 for missing answer tags, 0.1 for tagged but invalid or incorrect expressions, and 1.0 for fully correct expressions. We report rollout accuracy, defined as the fraction of individual completions with reward 1.0, as well as mean reward, prompt pass@16, all-zero prompt rate, all-correct prompt rate, and pass@k curves. The Heuristic Curriculum RLOO checkpoint is also re-evaluated under the same 16-sample protocol for a fair comparison.

4.2 Implementation Details and Hyperparameters

For SFT, we train Qwen2.5-0.5B with response-token-only cross-entropy, masking prompt tokens so that the loss is computed only on the target solution response. The model is trained for 6 epochs with

Table 1: Fixed local evaluation on 50 test prompts with 16 sampled completions per prompt. Rollout accuracy, Pass@16, and all-zero rate are percentages; reward is the mean scalar verifier reward.

Method	Rollout Acc.	Reward	Pass@16	All-zero
IPO	36.25	0.414	76.0	24.0
RLOO	48.25	0.533	74.0	26.0
Heuristic Curriculum RLOO	29.88	0.361	76.0	24.0
Self-Improvement RLOO	55.25	0.586	**80.0**	20.0
Phased Adaptive from SFT	**59.25**	**0.604**	72.0	28.0

Table 2: Additional diagnostics for the same fixed evaluation. Wilson 95% intervals are reported in percentage points. All-correct is the percentage of prompts for which all samples are correct, and mean correct is the average number of correct samples per prompt.

Method	Rollout 95% CI	Pass@16 95% CI	All-correct	Mean correct
IPO	[33.0, 39.6]	[62.6, 85.7]	0.0	5.80
RLOO	[44.8, 51.7]	[60.4, 84.1]	4.0	7.72
Heuristic Curriculum RLOO	[26.8, 33.1]	[62.6, 85.7]	0.0	4.78
Self-Improvement RLOO	[51.8, 58.7]	[67.0, 88.8]	8.0	8.84
Phased Adaptive from SFT	[55.8, 62.6]	[58.3, 82.5]	20.0	9.48

learning rate 5×10^{-5}, effective batch size 64, cosine warmup scheduling, and gradient checkpointing. IPO is trained for 1 epoch with $\beta = 0.1$, learning rate 5×10^{-6}, and summed response-token sequence log-probabilities against a frozen SFT reference model. Baseline RLOO is trained for 100 steps. Each step samples 8 responses for each of 128 prompts, producing 1024 sequences per update. RLOO uses learning rate 10^{-5}, constant scheduling, entropy and KL coefficients of 0.001, and evaluation every 10 steps with 16 rollouts per prompt.

The curriculum and self-improvement variants keep the same verifier and RLOO objective, but modify the training distribution or selected update groups. Heuristic Curriculum RLOO sorts training prompts by a scalar difficulty estimate and trains over three cumulative stages: the easiest third, easiest two thirds, and full dataset. Self-Improvement RLOO mixes real prompts with guaranteed-solvable synthetic prompts generated from real number sets using $+$, $-$, and $\times$. It uses a 5-step fully synthetic warmup and then anneals the synthetic ratio from 0.75 to 0.15 over 60 steps. Phased Adaptive Mixed-Group Filtering further classifies rollout groups as all-zero, mixed, or all-one. After warmup, it prioritizes mixed groups, retains a small fraction of all-correct real groups, drops all-correct synthetic groups by default, and adds fallback real prompts when too few groups remain. This variant anneals the synthetic ratio from 0.75 to 0.05 over 40 steps, uses 3–4 number synthetic prompts, and maintains a hard-prompt buffer for diagnostic resampling.

5 Results

Quantitative comparison. Table 1 reports the main fixed-evaluation results, and Figure 2 shows rollout accuracy with Wilson confidence intervals. IPO obtains 36.25% rollout accuracy and 76.0% Pass@16, providing the offline preference-optimization baseline. Standard RLOO improves rollout accuracy to 48.25% and mean reward to 0.533, showing that online verifier-based updates improve single-sample correctness over IPO. However, RLOO has slightly lower Pass@16 than IPO, indicating that the gain comes from more reliable samples on some prompts rather than broader prompt coverage.

Heuristic Curriculum RLOO performs worst in sample-level reliability, reaching only 29.88% rollout accuracy and the lowest mean reward, 0.361. Its Pass@16 remains 76.0%, matching IPO, but its mean correct count falls to 4.78. This suggests that the static easy-to-hard curriculum can occasionally find correct answers under repeated sampling, but does not make individual generations reliable. In contrast, Self-Improvement RLOO improves both sample-level and prompt-level performance, reaching 55.25% rollout accuracy and the best Pass@16, 80.0%. It also has the lowest all-zero rate, 20.0%, showing that it solves the largest fraction of prompts at least once. Phased Adaptive from SFT achieves the best rollout accuracy, 59.25%, and the highest mean reward, 0.604, but its Pass@16

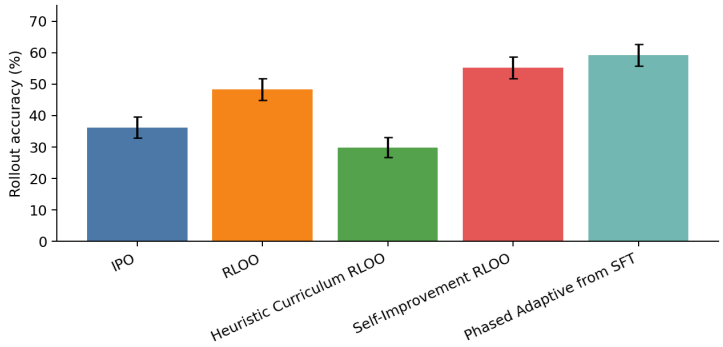

Figure 2: Rollout accuracy with Wilson 95% confidence intervals over 800 sampled completions per method. Self-improvement improves over standard RLOO, while the fixed heuristic curriculum degrades sample-level reliability.

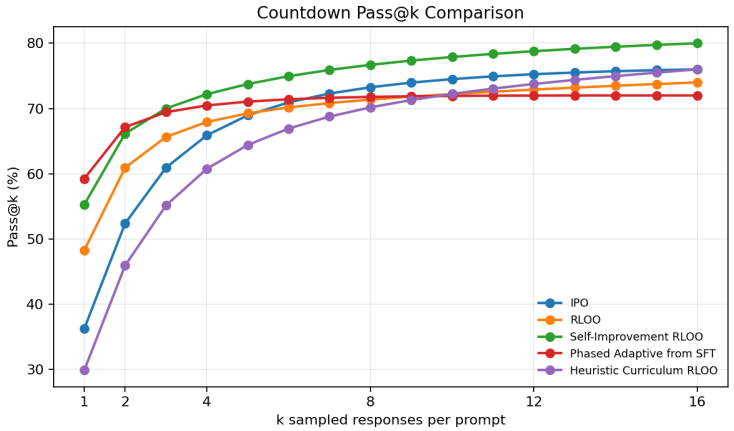

Figure 3: Pass@k curves under the fixed 16-sample evaluation. Phased Adaptive from SFT performs best at low k, while Self-Improvement RLOO achieves the strongest high-k coverage.

is lower at 72.0%. Thus, Self-Improvement RLOO provides the best coverage under multi-sample inference, while Phased Adaptive from SFT provides the strongest single-sample reliability.

Sampling behavior and per-prompt reliability. Figure 3 and Table 2 clarify why rollout accuracy and Pass@16 differ. Rollout accuracy measures correctness over individual samples, while Pass@16 measures whether at least one of 16 samples solves a prompt. Phased Adaptive from SFT has the highest all-correct rate, 20.0%, and the highest mean correct count, 9.48, indicating that it often generates consistently correct answers once a prompt is within its solved region. However, its 28.0% all-zero rate shows that it leaves more prompts unsolved. Self-Improvement RLOO has lower rollout accuracy than Phased Adaptive from SFT, but it reduces the all-zero rate to 20.0%, explaining its stronger Pass@16. Heuristic Curriculum RLOO shows the opposite behavior: its Pass@16 recovers under repeated sampling, but its low rollout accuracy and zero all-correct prompts indicate that its correct answers are sparse and unstable.

Figure 4 provides the same interpretation at the prompt level. IPO has moderate prompt coverage but no all-correct prompts, suggesting weak consistency. RLOO increases the number of correct samples per solved prompt, improving rollout accuracy. Heuristic Curriculum RLOO fails to increase reliability, despite maintaining broad stochastic coverage. Self-Improvement RLOO shifts more prompts away from the zero-correct bin, supporting its role as a coverage-improving curriculum. Phased Adaptive from SFT moves more solved prompts toward the high-correct-count and all-correct bins, but also leaves more prompts with zero correct samples.

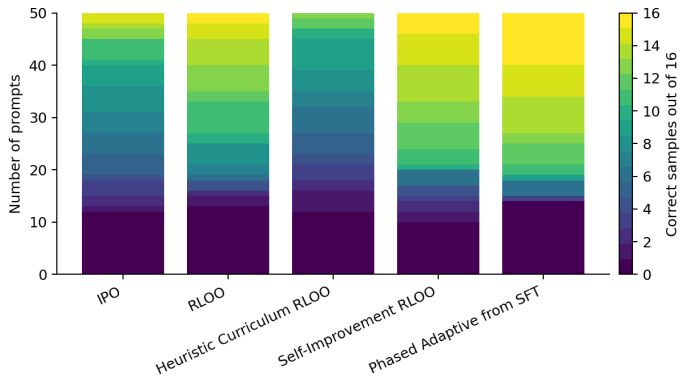

Figure 4: Distribution of the number of fully correct samples per prompt out of 16. Self-Improvement RLOO reduces the zero-correct bin, while Phased Adaptive from SFT creates more all-correct prompts but also leaves more prompts unsolved.

Qualitative analysis. A shared test example illustrates the quantitative trend. For the prompt with numbers $[70, 7, 15, 84]$ and target 79, IPO and standard RLOO produce no fully correct rollout among the sampled completions. Self-Improvement RLOO, however, finds the valid expression

$$(70/7 + 84) - 15 = 79.$$

This example is consistent with the aggregate results: self-improvement does not merely improve average reward, but increases the chance of discovering at least one valid solution under multi-sample inference. Qualitatively, the methods therefore exhibit different failure modes. IPO lacks online verifier feedback, RLOO improves sample-level correctness but not coverage, Heuristic Curriculum RLOO suffers from unreliable individual generations, Self-Improvement RLOO improves coverage by densifying early reward signal, and Phased Adaptive from SFT improves consistency on solved prompts while sacrificing some prompt coverage.

6 Discussion

The results suggest that curriculum learning improves verifier-based RL only when it increases informative reward variation. In RLOO, mixed rollout groups are most useful because they contain both correct and incorrect samples for the same prompt. All-zero groups provide little learning signal, while all-correct groups provide little contrast. Thus, effective curricula should target prompts near the model's current learnability boundary, rather than simply ordering examples from easy to hard.

This explains why the Heuristic Curriculum RLOO underperforms. Its static difficulty score does not adapt to the policy's actual competence, so it can still produce rollout groups that are either too hard or too easy. Although its 76.0% Pass@16 shows that it sometimes finds correct answers under repeated sampling, its 29.88% rollout accuracy indicates poor single-sample reliability.

Self-Improvement RLOO is more effective because it adds generated solvable stepping-stone prompts, making early rewards denser and increasing the chance of mixed groups. This leads to the best Pass@16 and the lowest all-zero rate, showing stronger prompt-level coverage. Phased Adaptive Self-Improvement instead achieves the highest rollout accuracy and all-correct rate, but its lower Pass@16 and higher all-zero rate show that it improves consistency on solved prompts at the cost of broader coverage.

Overall, the results indicate that useful curricula for verifier-based RL should be adaptive to policy behavior and should balance coverage with reliability. Future work should test larger models and harder Countdown variants, tune the synthetic-data and filtering schedules, and design curricula that explicitly target mixed-reward groups while preserving broad prompt coverage.

7 Conclusion

We studied verifier-based reinforcement learning for Countdown arithmetic reasoning and evaluated several curriculum-oriented extensions to RLOO. Standard RLOO improves over IPO, confirming that online optimization with rule-based rewards is effective for this task. However, the fixed heuristic curriculum underperforms, showing that static easy-to-hard ordering alone does not guarantee useful RL updates. Self-Improvement RLOO achieves the best Pass@16 by adding generated solvable stepping-stone prompts that improve prompt-level coverage under sampling. Phased Adaptive Self-Improvement from SFT achieves the best rollout accuracy, but with lower prompt coverage, indicating stronger single-sample reliability on a narrower set of prompts. Overall, the results suggest that curriculum learning helps verifier-based RL when it increases the density of learnable rewarded trajectories. The most promising direction is to combine the coverage of self-improvement with the reliability of adaptive mixed-group filtering, while maintaining broad exposure to the original task distribution.

8 Team Contributions

All work for this individual project was completed by Yuyan Wu.

Changes from Proposal The original proposal focused on adding curriculum learning to RLOO by ordering Countdown prompts according to difficulty. During experimentation, we found that this simple heuristic curriculum underperformed standard RLOO, so the project shifted toward comparing a small set of curriculum designs that directly address sparse verifier rewards. In the final experiments, we evaluate IPO and standard RLOO as baselines, and compare three RLOO-based curriculum variants: Heuristic Curriculum RLOO, Self-Improvement RLOO with synthetic solvable prompts, and Phased Adaptive training from SFT. This changes the extension from a single fixed curriculum method into a focused empirical study of which curriculum mechanisms provide useful reward variation during verifier-based RL fine-tuning.

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. arXiv:2402.14740 [cs.LG]

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2026. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. 700–719.

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.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. 2025. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970* (2025).

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*. PMLR, 482–495.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of Search: Learning to Search in Language. arXiv:2404.03683 [cs.CL]

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A General Theoretical Paradigm to Understand Learning from Human Preferences. arXiv:2310.12036 [cs.LG]

Jiayi Pan. 2025. TinyZero. `https://github.com/Jiayi-Pan/TinyZero`.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. arXiv:2305.20050 [cs.LG]

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3732–3740.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research* 21, 181 (2020), 1–50.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664* (2020).

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. arXiv:2305.18290 [cs.LG]

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520* (2025).

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2017. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407* (2017).

Shobhita Sundaram, John Quan, Ariel Kwiatkowski, Kartik Ahuja, Yann Ollivier, and Julia Kempe. 2026. Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability. *arXiv preprint arXiv:2601.18778* (2026).

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving Math Word Problems With Process- and Outcome-Based Feedback. arXiv:2211.14275 [cs.LG]

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. 2025. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710* (2025).

Xuan Zhang, Ruixiao Li, Zhijian Zhou, Long Li, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. 2025. Count Counts: Motivating Exploration in LLM Reasoning with Count-based Intrinsic Rewards. arXiv:2510.16614 [cs.AI]