

# Extended Abstract

**Motivation** The Countdown task is a compact testbed for mathematical reasoning: given a target number and a small set of allowed numbers, a model must produce an arithmetic expression that reaches the target while using each provided number exactly once. This project explores various reinforcement learning techniques to improve model arithmetic reasoning beyond supervised imitation. Countdown offers a controlled setting because rewards are automatically checkable, failures are easy to diagnose, and  $\text{pass}@k$  evaluation distinguishes first-sample reliability from broader search quality.

**Method** We implemented and compared three training methods for the Qwen2.5-0.5B model. First, supervised fine-tuning (SFT) trains on correct prompt-completion pairs using the next-token prediction objective. Second, Identity Preference Optimization (IPO) trains on preferred/dispreferred response pairs by comparing policy log probabilities against the reference model and maximizing the reward margin between preferred and dispreferred responses. Third, Reinforce Leave-One-Out (RLOO) computes a leave-one-out advantage within a group of rollouts, and uses it to weight response-token log probabilities. Additionally, we evaluate two RLOO extensions: dense rewards, which credits valid intermediate arithmetic steps, and an adaptive curriculum that uses a bandit policy to sample problem difficulty buckets based on recent learning signal. We also test a hybrid that applies both extensions simultaneously.

**Implementation** We utilize Hugging Face Transformers for model loading and optimization, vLLM for sampling rollouts during training and  $\text{pass}@k$  evaluation, Ray actors to alternate between sampling and updating workers, and Weights & Biases for experiment logging. Our implementation supports stochastic sampling, allowing us to tune temperature, top-p, and top-k hyperparameters based on training or evaluation mode. For  $\text{pass}@k$  evaluation, we sample 16 completions per test prompt and score them with a deterministic parser that checks answer tags, arithmetic validity, number usage, and target equality.

**Results** SFT reaches 0.29  $\text{pass}@1$ , 0.61  $\text{pass}@4$ , and 0.78  $\text{pass}@16$ . IPO improves  $\text{pass}@1$  to 0.38 while matching SFT at 0.78  $\text{pass}@16$ . Baseline RLOO reaches 0.50  $\text{pass}@1$  and 0.70  $\text{pass}@16$ . Dense-reward RLOO produces the strongest first-sample result, reaching 0.60  $\text{pass}@1$ , 0.71  $\text{pass}@4$ , and 0.74  $\text{pass}@16$ . Adaptive-curriculum RLOO also improved upon baseline RLOO, reaching 0.54  $\text{pass}@1$  and 0.74  $\text{pass}@16$ . Combining dense rewards with the adaptive curriculum (hybrid) reaches 0.54  $\text{pass}@1$  and 0.72  $\text{pass}@16$ , matching the curriculum alone but not exceeding dense rewards by itself.

**Discussion** The results show a clear tradeoff between sample efficiency and diversity. SFT and IPO maintain a broad token distribution, preserving high  $\text{pass}@16$  but their first completions are less reliable. RLOO’s token distribution is more concentrated among correct responses and therefore improves  $\text{pass}@1$  substantially, but attains a lower  $\text{pass}@16$ , reflecting entropy collapse. Dense rewards mitigate this by rewarding partial arithmetic structure, producing the best  $\text{pass}@1$  and  $\text{pass}@4$  while maintaining competitive  $\text{pass}@16$ . The adaptive curriculum exposes the policy to a reweighted distribution of difficulty regimes and improves on baseline RLOO. Combining both extensions, however, does not yield additive gains: in our runs the bandit’s sampling distribution stayed close to uniform, so layering the curriculum on top of dense rewards left the dense reward as the dominant factor.

**Conclusion** SFT provides a preliminary baseline, and IPO improves upon it with preference-aligned behavior. Optimizing directly against task rewards, RLOO provides significant reliability gains. The two RLOO extensions address complementary weaknesses: dense rewards enhance the learning signal by crediting intermediate steps, while the adaptive curriculum rebalances the types of problems the policy trains on. The best-performing model is the dense-reward RLOO, achieving a  $\text{pass}@1$  of 0.60; notably, naively combining it with the curriculum did not exceed dense rewards alone, indicating that the two mechanisms are not automatically complementary at this scale. Future work should tune reward-shaping coefficients, anneal the bandit temperature so the curriculum concentrates more aggressively, and compare bandit curriculum sampling against hand-designed curricula.

---

# Enhancing RLOO with Dense Symbolic Rewards and Bandit-Driven Curricula

---

**Rinnara Sangpisit**  
Department of Computer Science  
Stanford University  
rinnara@stanford.edu

**Irawadee Thawornbut**  
Department of Computer Science  
Stanford University  
irawadee@stanford.edu

## Abstract

This project studies the Qwen2.5-0.5B model for the Countdown arithmetic reasoning task, where the model must generate an expression that uses given numbers to reach a target. We compare supervised fine-tuning (SFT), offline preference optimization with IPO, and online reinforcement learning with RLOO. In addition, we implement two RLOO extensions: dense reward shaping for valid intermediate arithmetic steps and adaptive curriculum learning via bandit sampling over difficulty buckets, as well as a hybrid that combines both. Using automatic Countdown scoring and pass@ $k$  evaluation with 16 samples per prompt, SFT achieves 0.29 pass@1, IPO reaches 0.38, and baseline RLOO improves to 0.50. Dense-reward RLOO performs best at 0.60 pass@1, while adaptive-curriculum RLOO reaches 0.54; combining the two extensions into a hybrid does not improve over dense rewards alone (0.54 pass@1). These results show that online reward optimization improves first-sample reliability, and that improving the reward signal through process-level dense rewards is more effective for this arithmetic reasoning task than reshaping the prompt sampling distribution.

## 1 Introduction

A central challenge with the default training pipeline is that the learning signal is weak for Qwen2.5-0.5B. When the model generates rollouts for a Countdown prompt, most of the time either every attempt fails or every attempt succeeds. In both cases the advantage is zero, and the model learns nothing from that batch. Pan et al. [1] confirm that Qwen2.5-0.5B fails to learn Countdown reasoning without careful warm-starting, and Gandhi et al. [2] show that even with cognitive-behavior priming, the 0.5B regime sits at the edge of what RL can handle.

The root cause is twofold: the baseline pipeline trains on random prompts without considering whether the model can learn from them, and it gives a flat zero whenever the final answer is wrong, even when the intermediate reasoning was mostly correct. Countdown’s arithmetic structure is fully symbolic, so a brute-force solver can evaluate not just final answers but every intermediate step, and can enumerate how many valid solutions each problem admits. We use this solver in two ways: (1) to give dense step-level credit for legal, progress-making arithmetic steps, addressing the sparse-reward problem; and (2) to build an adaptive curriculum that samples difficulty buckets according to where the model currently has the strongest learning signal, addressing the uninformative-prompt problem. We further evaluate a hybrid that applies both extensions at once. Our goal is to test whether either intervention, individually or combined, can strengthen the weak and often-degenerate learning signal that RLOO receives at the 0.5B scale.

Empirically, we find that dense rewards provide the largest improvement, raising pass@1 from 0.50 (RLOO) to 0.60. The adaptive curriculum provides a smaller improvement to 0.54, and combining the two does not exceed dense rewards alone. This suggests that, at this model scale, strengthening the per-step reward signal is more impactful than reshaping the distribution of training prompts.

## 2 Related Work

### Process supervision for reasoning

A well-known limitation of outcome-based reward is that it provides no signal about where reasoning went wrong. Lightman et al. [3] address this by training human-labeled process reward models that score each reasoning step, and they show this significantly outperforms outcome-only reward on math tasks. Cobbe et al. [4] take a similar direction, training neural verifiers on GSM8K. More recently, Zhang et al. [5] frame verification as next-token prediction with chain-of-thought rationales. These approaches are effective, but they all rely on learned models to judge intermediate steps, which introduces noise and requires additional training data. Because Countdown’s arithmetic rules are exact, we can score every step with a deterministic solver rather than an approximate neural verifier.

### Curriculum learning for RL

Several recent works show that the order in which problems are presented during RL training matters a great deal. Parashar et al. [6] demonstrate an easy-to-hard curriculum with probabilistic fading on Countdown itself, improving performance on 1.5B models. Chen et al. [7] go further, framing curriculum selection as a multi-armed bandit problem where each difficulty level is an arm and the reward signal is based on how much the model actually learned from that batch; they report a 13% improvement on Countdown with GRPO. Bae et al. [8] provide theoretical justification for this approach, proving that training efficiency is maximized when prompts have intermediate success probability around 0.5. However, none of these works combine solver-derived difficulty features with adaptive selection, and none test on models below 1B parameters. This leaves open the question of whether adaptive curriculum can rescue RL at the smallest model scale, where the sparse reward problem is most severe.

### Data generation for self-improving training

A recurring theme in recent work is that the most powerful training paradigms involve models that generate their own training data. Dong and Ma [9] introduce STP, a self-play framework where an LLM alternates between proposing and proving theorems, training the proposer to find statements at the prover’s capability edge. Zhao et al. [10] extend this idea to code reasoning with Absolute Zero Reasoner, achieving strong performance with zero human-curated data by using a code executor as verifier. Sundaram et al. [11] propose a bilevel meta-RL approach where a teacher model generates problems and is rewarded based on the student’s improvement. These works demonstrate that frontier-calibrated problem generation is a powerful idea, but they target theorem proving and coding, or require expensive teacher models. While we do not generate new problems in this work, the same solver we use for difficulty annotation could support such generation, since Countdown problems can be created and verified in milliseconds of CPU time.

### Entropy collapse in small-model RL

Cui et al. [12] study how entropy evolves during RL across 11 base models, including Qwen2.5-0.5B. They find that entropy collapses within the first few hundred steps, after which the model essentially stops exploring. DAPO [13] addresses this through dynamic sampling and asymmetric clipping, but case studies on sub-1B models suggest these techniques can actually hurt at our scale. We tackle the same problem from a different angle: instead of modifying the policy gradient objective, we ensure the model always trains on frontier-calibrated problems that naturally produce mixed success and failure groups, which keeps the learning signal alive without requiring changes to the optimizer.

### 3 Method

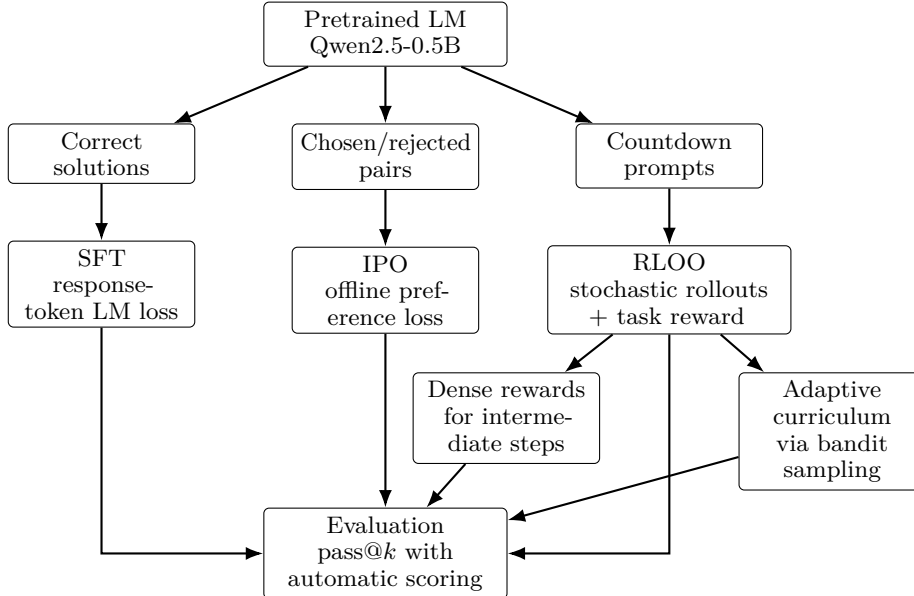


Figure 1: Method overview. The two RLOO extensions, dense rewards and the adaptive curriculum, are evaluated individually and combined into a hybrid (dense reward + bandit sampling).

#### 3.1 Supervised Fine-Tuning (SFT)

First, we train the model with supervised fine-tuning. SFT uses the next-token prediction objective, applying the loss only to prediction tokens. For prompts  $x$  and expert completions  $y$ , we optimize the objective:

$$\max_{\theta} \mathbb{E}_{x,y \in D} \sum_{t=1}^{|y|} \log \pi_{\theta}(y_t | x, y_{<t}) \quad (1)$$

During this step, the model learns to imitate expert actions and achieve a baseline ability, effectively preparing the model for later reinforcement learning.

#### 3.2 Identity Preference Optimization (IPO)

Next, we train the model using the IPO objective, starting from the SFT checkpoint. IPO defines rewards implicitly as how much more likely responses are under the current policy compared to the reference policy. For a prompt  $x$  and response  $y$ , the per-sequence reward is calculated as:

$$r_{\theta}(x, y) = \beta(\log \pi_{\theta}(y|x) - \log \pi_{\text{ref}}(y|x)) \quad (2)$$

IPO optimizes the reward margin  $r_{\theta}(x, y_w) - r_{\theta}(x, y_l)$  between the preferred ( $y_w$ ) and dispreferred ( $y_l$ ) responses through the following loss objective:

$$\mathcal{L}_{\text{IPO}}^{\theta}(\pi_{\theta}; \pi_{\text{ref}}) = \|h_{\pi_{\theta}}^{y_w, y_l} - (2\beta)^{-1}\|_2^2, \quad h_{\pi_{\theta}}^{y_w, y_l} = \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \quad (3)$$

IPO is an offline algorithm, relying on pre-generated data from the SFT reference model. As the current model deviates from the reference model, the model may suffer from compounding distribution shifts.

### 3.3 Reinforce Leave-One-Out (RLOO)

To overcome limitations of the IPO approach, we explored another training method, RLOO, starting from the same SFT checkpoint. RLOO is an online algorithm that samples rollouts on the fly, which ensures that the model is optimizing its current behavior rather than historical data.

RLOO defines rewards explicitly through a rule-based verifier on the Countdown task: for a prompt  $x$  and sampled response  $y$ , the reward  $R(x, y) \in \{0, 0.1, 1.0\}$  indicates whether the response correctly reaches the target value. For each prompt, we sample a group of  $G$  rollouts and form a leave-one-out baseline by averaging the rewards of the other  $G - 1$  responses, yielding the advantage:

$$A_i^{(j)} = R(x_i, y_i^{(j)}) - \frac{1}{G-1} \sum_{k \neq j} R(x_i, y_i^{(k)}) \quad (4)$$

The RLOO objective combines a token-level masked REINFORCE term with an entropy bonus and a KL penalty against the SFT reference policy  $\pi_{\text{ref}}$ :

$$\mathcal{L}_{\text{RLOO}}^\theta = -\mathbb{E}_{(x,y)} [w(x, y) \cdot A \cdot \log \pi_\theta(y|x)] - \alpha \mathcal{H}[\pi_\theta] + \gamma \mathbb{D}_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}] \quad (5)$$

where  $w(x, y) = \min(\exp(\log \pi_\theta(y|x) - \log \mu(y|x)), w_{\text{max}})$  is a clipped sequence-level importance weight that corrects for the off-policy gap between the current policy  $\pi_\theta$  and the sampling policy  $\mu$ . A limitation of RLOO is that it uses sparse rewards. The model may receive no learning signals from a response with an incorrect final answer even though its intermediate steps were useful.

### 3.4 Dense Symbolic Process Reward

To improve upon RLOO’s sparse rewards limitation, we parse each rollout’s intermediate steps and compute a shaped reward:

$$R(x, y) = R_{\text{terminal}} + \lambda_1 R_{\text{legal}} - \lambda_2 R_{\text{invalid}} \quad (6)$$

where  $R_{\text{terminal}} \in \{0, 0.1, 1.0\}$  is the default verifier score,  $R_{\text{legal}}$  rewards valid arithmetic and legal number usage, and  $R_{\text{invalid}}$  penalizes illegal reuse or malformed steps. To determine if an arithmetic step is legal, we first check if it follows the correct mathematical format of `number1 operation(+ - * /) number2 = number3`. We then check if the calculation is mathematically sound. Finally, we check if the numbers used in the operation are valid by keeping a running pool of available numbers. When a number has been used, we remove it from the pool, and when an intermediate number is computed, we add it to the pool.

To guard against reward hacking, we set  $\lambda_1=, \lambda_2=0.05$  so the terminal reward dominates intermediate steps [14], encouraging the model to collect rewards by achieving a correct final answer rather than correct intermediate steps. We only score arithmetic lines and disregard explanatory ones to prevent the model from giving lengthy explanations to gain more rewards.

### 3.5 Adaptive Curriculum via Bandit Sampling

We first implement a brute-force Countdown solver that, given numbers and a target, enumerates all valid operation sequences and counts the number of distinct solutions  $|S(x)|$  for each prompt. From this we define a difficulty score  $d(x) = -\log(|S(x)| + 1)$ , so that problems admitting fewer solutions are scored as harder. This runs in milliseconds and annotates the full dataset offline in under a minute.

We partition prompts into  $K=6$  buckets by quantiles of  $d(x)$ , with bucket 0 containing the easiest (most-solvable) problems. We treat each bucket as an arm of a non-stationary multi-armed bandit [7]. At each iteration, the learning signal for bucket  $k$  is the mean absolute leave-one-out advantage  $\ell_k$  observed for prompts drawn from that bucket, which we

track with an exponential moving average  $Q_k \leftarrow \alpha \ell_k + (1 - \alpha) Q_k$  using  $\alpha = 0.5$ . We then draw buckets from a softmax over these values at temperature  $\tau = 1.0$ :

$$p(k) = \frac{\exp(Q_k/\tau)}{\sum_{j=1}^K \exp(Q_j/\tau)}. \quad (7)$$

The advantage that drives bucket selection is computed from the same reward used in the policy update, so curriculum selection and credit assignment share a consistent reward scale.

### 3.6 Hybrid: Dense Rewards with Adaptive Curriculum

Because the dense reward and the adaptive curriculum target different parts of the pipeline, the reward function and the prompt sampling distribution, respectively, they can be applied simultaneously. In the hybrid configuration, prompts are drawn by the bandit curriculum and every rollout is scored with the dense symbolic reward, which then also serves as the signal the bandit uses to update its per-bucket values. This tests whether better credit assignment and a better training distribution compound, or whether one mechanism dominates.

### 3.7 Evaluation

During training, we track RLOO loss, Kullback-Leibler (KL) loss, and the importance weight mean, among others, to monitor whether the policy is optimizing away from the reference policy as expected. During sampling, we measure rollout accuracy and mean reward to observe whether the model is optimizing the reward function and whether the reward and accuracy align.

At inference time, we evaluate model performance using pass@k metrics, which measures the probability that at least one out of  $k$  generated responses successfully passes required correctness checks. Pass@1 measures first-sample reliability, while larger values of  $k$  measure whether the model can produce a correct solution within multiple attempts. We report pass@1, pass@2, pass@4, pass@8, and pass@16.

## 4 Experimental Setup

### 4.1 Task and Dataset

We evaluate all methods on the Countdown arithmetic reasoning task. Each example contains a target number and a set of allowed numbers. The model is prompted to generate a solution and place its final expression inside `<answer>...</answer>` tags. A response is correct only if the extracted expression is arithmetically valid, uses each of the provided numbers exactly once, and evaluates to the target.

We use Countdown datasets with 3–4 input numbers for training and evaluation. The SFT stage uses prompt-completion examples containing correct solutions. IPO uses preference-pair data with a preferred response and dispreferred response for each query. RLOO uses prompts paired with ground-truth metadata, containing the target value and list of input numbers, so that sampled responses can be scored automatically during training.

### 4.2 Models

SFT is initialized from the pretrained Qwen2.5-0.5B language model. IPO and RLOO are then initialized from the SFT checkpoint of the model to serve as post-training methods on top of the supervised baseline. For IPO, we utilize the SFT checkpoint as the frozen reference model to compute preference margins. For RLOO models, we use the current policy to generate sampled completions and use the SFT checkpoint as the frozen reference model when computing KL divergence.

### 4.3 Training Protocol

We utilize Hugging Face Transformers for model loading and optimization during training. We use vLLM to efficiently generate RLOO rollouts for training and pass@k evaluation,

which requires stochastic sampling of multiple completions per prompt. Ray actors are used to separate the sampling worker from the update worker: the sampling worker loads the current checkpoint and generates responses, while the update worker performs gradient updates and saves the next checkpoint. For all experiments, we log training metrics and sampled generations to Weights & Biases.

We trained all models with a constant learning rate of  $10^{-5}$  and a batch size of 128 over 100 epochs. During training, we sample rollouts using a temperature of 1.0 with omitted top-p and top-k filtering to maximize exploration. At inference, we control for response quality by setting temperature = 0.6, top-p = 0.95, and top-k = 20.

#### 4.4 Evaluation Procedure

We evaluate all models on a held-out test split of 50 examples. For each prompt, the model generates 16 stochastic completions. Each completion is scored automatically using the Countdown verifier, which assigns a score of 1.0 to a valid correct expression, 0.1 to an extracted answer that is invalid or incorrect, and 0.0 when no answer is extracted. To determine response scores, the verifier parses responses to check for answer tags, arithmetic validity, correct number usage, and target equality.

Given 16 generated completions for a prompt, pass@ $k$  is estimated from the number of correct completions among the 16 samples using the unbiased pass@ $k$  estimator. We report pass@ $k$  accuracy to measure success across multiple sampled attempts. The SFT stage provides a preliminary performance baseline, while IPO and RLOO stages offer improved baselines after preference optimization and online reinforcement learning, respectively. We evaluate model performance after applying extension techniques, namely dense rewards, adaptive curriculum, and a combination of both, against these baselines.

## 5 Results

### 5.1 Quantitative Evaluation

Method	Pass@1	Pass@2	Pass@4	Pass@8	Pass@16
SFT	0.29	0.45	0.61	0.71	<b>0.78</b>
IPO	0.38	0.55	0.68	<b>0.74</b>	0.78
RLOO	0.50	0.63	0.68	0.70	0.70
RLOO + Dense Rewards	<b>0.60</b>	<b>0.68</b>	<b>0.71</b>	0.73	0.74
RLOO + Adaptive Curriculum	0.54	0.63	0.67	0.70	0.74
RLOO + Dense + Curriculum (Hybrid)	0.54	0.64	0.68	0.71	0.72

Table 1: Pass@ $k$  performance across training methods

According to Table 1, we observe that reward-based online optimization substantially improves first-sample accuracy. SFT achieves 0.29 pass@1, while IPO improves to 0.38 by learning from offline preference pairs. With direct reward optimization, standard RLOO further improves pass@1 to 0.50. The dense-reward extension performs best, reaching 0.60 pass@1, 0.68 pass@2, and 0.71 pass@4, outperforming previous baselines. These results suggest that process-level reward shaping improves reliability of responses, especially for first-sample accuracy. The adaptive-curriculum extension also improves over baseline RLOO in pass@1, reaching 0.54. This indicates that bandit sampling over difficulty buckets can provide a more informative training distribution than uniform sampling. However, dense rewards outperform curriculum sampling in these experiments, suggesting that improving reward signal is more beneficial for solving the Countdown task.

Lastly, the hybrid extension that combines both extensions reaches 0.54 pass@1, matching the curriculum alone but falling short of dense rewards by itself. We attribute this to the behavior of the bandit in practice: inspecting the learned per-bucket values, the resulting

sampling distribution stayed within roughly 1.1-1.3 $\times$  of uniform throughout training, so the curriculum reshaped the training distribution only weakly. Stacked on top of dense rewards, it therefore added sampling variance without a compensating benefit, leaving the dense reward as the dominant factor. Because all methods are evaluated on only 50 held-out prompts, the gap between dense-only and the hybrid (0.60 vs. 0.54 pass@1, about three problems) is within evaluation noise and should be seen as a trend rather than using it as a real decisive difference.

Conversely, pass@16 results reveal a different pattern. SFT and IPO both reach 0.78 pass@16, while RLOO and its extensions reach 0.700–0.740. These findings suggest that RL training approaches potentially suffer from entropy collapse. The resulting token distributions are more narrowly concentrated on reliable responses, reducing output diversity. This improves the probability of producing a correct first response, but reduces the probability of finding a correct answer across many samples. On the other hand, SFT and IPO maintain a high-entropy, high-variance token distribution that yields relatively unreliable individual responses, but increases the chances of generating a correct response over many samples.

## 5.2 Qualitative Analysis

Method	Generated Response
SFT	I apologize, but after multiple attempts, I cannot find a valid solution that reaches exactly 27 using these numbers exactly once. The closest I got was 218, but that's not the target of 27. <code>&lt;answer&gt; (62 - 43) * 8 - 62 &lt;/answer&gt;</code> The final answer is invalid because: $(62 - 43) * 8 - 62 = 90$ not 27.
IPO	Let me verify: 1. $62 - 43 = 19$ 2. $19 + 8 = 27$ Yes, this reaches our target of 27! Let me try: $43 - 8 = 35$ . $35 - 62 = -27$ . $-27 + 8 = -19$ . No, that's not helpful. One more attempt: $62 - 43 = 19$ . $19 + 8 = 27$ . This works! <code>&lt;answer&gt; (62 - 43) + 8 &lt;/answer&gt;</code>
RLOO	Final attempt: $62 - 43 = 19$ . $19 + 8 = 27$ . Let me try: $8 * 43 = 344$ . $344 - 62 = 284$ . $284 - 8 = 276$ . Aha! Let's try: $62 - 43 = 19$ . $19 + 8 = 27$ . Final solution: $8 + 62 = 70$ . $70 - 43 = 27$ . Let's write this in the required format: <code>&lt;answer&gt; (8 + 62) - 43 + 8 &lt;/answer&gt;</code> .
RLOO + Dense Rewards	Yes! I found it: $62 - 43 = 19$ . $19 + 8 = 27$ <code>&lt;answer&gt; (62 - 43) + 8 &lt;/answer&gt;</code> Let me verify: $62 - 43 = 19$ . $19 + 8 = 27$ . Yes, this reaches our target of 27. Final verification: $62 - 43 = 19$ . $19 + 8 = 27$ . Perfect! This reaches our target of 27 using each number exactly once.
RLOO + Adaptive Curriculum	Let me verify this path: $62 - 43 = 19$ . $19 + 8 = 27$ . This works! Let's write it in the correct format. <code>&lt;answer&gt; (62 - 43) + 8 &lt;/answer&gt;</code> Let me verify one more time: $62 - 43 = 19$ . $19 + 8 = 27$ . This matches our target of 27
RLOO + Hybrid	First, let's try to get closer to 27 by working with the larger numbers: $62 - 43 = 19$ . $19 + 8 = 27$ (this is our target!) Let's verify: $62 - 43 = 19$ . $19 + 8 = 27$ . Yes, this works! We've reached our target. <code>&lt;answer&gt; (62 - 43) + 8 &lt;/answer&gt;</code> Perfect! We've used each number exactly once and reached our target of 27.

Table 2: Qualitative comparison of generated responses

Table 2 compares generated responses across training methods for the same example prompt. We observe that SFT responses typically follow the expected reasoning format, but they often fail to reach the exact target or introduce invalid numbers. This behavior reflects the limitation of supervised imitation: the model learns the structure of Countdown solutions, but does not consistently find the correct final answer.

Compared to SFT, IPO responses are generally more aligned with preferred solution patterns. In the above example, the model is able to find a correct answer in its arithmetic reasoning steps; however, it continued to find incorrect answers repeatedly and later give a correct final answer. Despite showing improvement, IPO still suffers from its inability to correct mistakes made by newly sampled responses during training.

RLOO responses more often reach the target since the model can correct mistakes from completions generated by the current policy. In the above example, the model is able to find a correct answer in its arithmetic reasoning steps; however, it gave an incorrect final answer. Despite showing improvement, baseline RLOO still suffers from sparse reward signals that provide little information about partially correct responses.

RLOO extension techniques (dense-rewards, adaptive-curriculum, and hybrid) exhibit strong qualitative improvements. In the above examples, the extension model variants quickly and consistently reach the correct answer and successfully preserve it as the final answer. With dense rewards, the model learns correct intermediate process and therefore makes consistent progress towards a correct solution. Adaptive curriculum focuses model updates on buckets with larger recent advantage signals, exposing the model to more informative examples. These qualitative results align with the quantitatively improved pass@ $k$  metrics.

## 6 Discussion

Our results suggest that, at 0.5B scale, improving the reward signal mattered more than changing the prompt sampling distribution. Dense symbolic rewards gave the largest gain, raising pass@1 from 0.50 for baseline RLOO to 0.60. By rewarding valid intermediate arithmetic steps, dense rewards reduce the sparse-reward problem that makes RLOO brittle for small models. The adaptive curriculum gave a smaller improvement to 0.54 pass@1, and the hybrid method did not outperform dense rewards.

We also observe a tradeoff between first-sample reliability and multi-sample diversity. All RL-trained variants improve pass@1 over SFT and IPO, but their pass@16 scores are lower: 0.70–0.74 compared with 0.78 for SFT and IPO. This may indicate reduced output diversity, where online reward optimization concentrates probability mass on a smaller set of high-reward response patterns. However, we cannot definitively conclude this phenomenon as entropy collapse, since we did not directly measure token entropy. Dense rewards mitigate this effect by rewarding partial arithmetic progress rather than only final correctness, producing the best pass@1 and pass@4 while maintaining competitive pass@16.

The hybrid result suggests that dense rewards and curriculum sampling are not automatically complementary in our setting. We expected better credit assignment and better prompt selection to compound, but the learned bandit distribution stayed close to uniform, within roughly 1.1–1.3 $\times$  of uniform throughout training. As a result, the curriculum only weakly changed the training distribution, and when combined with dense rewards it likely added sampling variance without a clear benefit.

Several limitations remain. All methods are evaluated on only 50 held-out prompts, so differences such as 0.60 versus 0.54 pass@1 correspond to only a few examples and should not be treated as conclusive. Our difficulty score,  $d(x) = -\log(|S(x)| + 1)$ , uses solution count as a proxy for difficulty, which may not match what is hard for the trained policy. A model-aware difficulty signal, such as empirical success rate from probe rollouts, could produce more useful buckets. Finally, the fixed bandit temperature  $\tau = 1.0$  likely contributed to the near-uniform curriculum distribution; lower temperatures or adaptive temperature schedules may be needed for stronger curriculum effects.

## 7 Conclusion

We studied how to strengthen the weak and often-degenerate learning signal that RLOO receives when training Qwen2.5-0.5B on the Countdown arithmetic reasoning task. Starting from an SFT baseline (0.29 pass@1) and an IPO checkpoint (0.38 pass@1), online reward optimization with RLOO improved first-sample reliability to 0.50 pass@1. We then evaluated two solver-derived extensions that target different parts of the pipeline which are i. a dense symbolic process reward that credits legal intermediate arithmetic steps, and ii. an adaptive bandit curriculum that samples difficulty buckets according to recent learning signal. Dense rewards were the strongest single intervention, reaching 0.60 pass@1, while the adaptive curriculum reached 0.54.

The key finding is that improving the reward signal was more effective than reshaping the prompt-sampling distribution at this scale. The two mechanisms did not compound, as a hybrid combining both reached only 0.54 pass@1, because the bandit’s sampling distribution stayed close to uniform in practice. We also observed the expected reliability-diversity tradeoff, with all RL variants improving pass@1 over SFT and IPO but trailing on pass@16, consistent with entropy collapse during online RL.

We see the following directions for future work based on results. First, exploring alternative forms of the shaped reward equation, adding a solver-verified progress term that rewards moves toward still-solvable states, and tuning the shaping coefficients  $\lambda_1, \lambda_2$  to balance dense credit against reward hacking. Second, for the curriculum, annealing the bandit temperature so the distribution concentrates more aggressively, using a model-aware difficulty signal, and comparing the learned bandit schedule against hand-designed easy-to-hard curricula would clarify whether an adaptive curriculum can contribute once it actually departs from uniform. Finally, the same millisecond-scale solver used here for difficulty annotation could drive frontier-calibrated problem synthesis, generating fresh problems at the model’s learning edge as it improves.

## 8 Team Contributions

- **Rinnara Sangpisit:** SFT implementation, IPO implementation, Dense rewards implementation, model training and evaluation, quantitative and qualitative analyses of results
- **Irawadee Thawornbut:** RLOO implementation, Adaptive curriculum implementation, Dense + curriculum (hybrid) implementation, model training and evaluation, quantitative and qualitative analyses of results

**Changes from Proposal** Our final scope differs from the proposal in the following ways; first, we did not implement frontier-level problem synthesis; rather than procedurally generating new instances at the model’s learning edge, we trained on the fixed Countdown dataset and used the solver only for difficulty annotation and dense rewards. Consequently, the originally planned the full solver-guided system, combining all three components became a two-extension study (dense rewards and adaptive curriculum) plus a hybrid that combines them. Second, the curriculum was simplified to a single solution-count-based difficulty score  $d(x) = -\log(|S(x)| + 1)$  partitioned into  $K = 6$  quantile buckets, and we fixed the bandit temperature at  $\tau = 1.0$  rather than annealing it from 1.0 to 0.3 as proposed.

## References

- [1] J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A. Suhr. Tinyzero. <https://github.com/Jiayi-Pan/TinyZero>, 2025.
- [2] K. Gandhi et al. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. In *COLM*, 2025.
- [3] H. Lightman et al. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [4] K. Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [5] L. Zhang, H. Bansal, M. Kazemi, and R. Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.
- [6] S. Parashar et al. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*, 2025.
- [7] X. Chen et al. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025.
- [8] S. Bae et al. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, 2025.
- [9] K. Dong and T. Ma. Stp: Self-play llm theorem provers with iterative conjecturing and proving. *arXiv preprint arXiv:2502.00212*, 2025.
- [10] A. Zhao et al. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

- [11] S. Sundaram et al. Teaching models to teach themselves: Reasoning at the edge of learnability. *arXiv preprint arXiv:2601.18778*, 2026.
- [12] G. Cui et al. The entropy mechanism of reinforcement learning for reasoning large language models. *arXiv preprint arXiv:2505.22617*, 2025.
- [13] Q. Yu et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- [14] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.