# Extended Abstract

**Motivation and Problem Statement** Large language models can improve their reasoning ability through reinforcement learning from verifiable rewards, but online RL fine-tuning is often inefficient when rewards are sparse. In tasks such as Countdown arithmetic, reward is sparse because high-reward rollouts require the model to produce a well-formed solution in which the arithmetic expression reaches the target number exactly. Malformed or incorrect rollouts provide little useful learning signal. Even after supervised fine-tuning (SFT), many sampled rollouts may still be invalid or incorrect, especially early in RL training. This creates a cold-start problem: the policy spends compute exploring prompts that produce little learning signal, while high-variance policy updates can destabilize the policy. This failure mode is not specific to Countdown. Any task with sparse, verifiable rewards, such as code generation, theorem proving, and tool use, can face the same early-training bottleneck. This project studies whether curriculum-based prompt ordering can make RL fine-tuning more stable and sample-efficient for language-model reasoning. Rather than aiming only for the highest final score, the goal is to understand when curriculum structure helps, when it fails, and what failure modes appear in small-scale online RL fine-tuning.

**Method and Novelty** I fine-tune with RLOO (REINFORCE Leave-One-Out) and compare three families of prompt ordering: (1) a baseline with no curriculum, (2) a static curriculum that orders prompts by operand count (3-operand before 4-operand problems), and (3) curricula that order prompts by an *SFT-estimated difficulty signal.* The novel component is the difficulty estimator: rather than relying on a hand-crafted heuristic (operand count) or a separately learned curriculum policy, I use the supervised fine-tuned checkpoint itself to score each prompt by its empirical pass rate over sampled rollouts, yielding a model-aware difficulty estimate that reflects what the policy can actually solve at initialization. Transition timing is itself a design axis: hard stage boundaries versus delayed transitions. Curriculum prompts are released in stages with increasing data fractions (50% / 75% / 100%) under delayed stage-transition schedules, so the policy is exposed to easier, higher-yield prompts before harder and full-data prompts are introduced. The approach requires no architecture changes and is a lightweight addition to a standard RLOO pipeline.

**Implementation Details and Headline Results** All runs fine-tune Qwen2.5-0.5B with RLOO on the asingh15/countdown_tasks_3to4 dataset and are evaluated by empirical pass@16 on a held-out 50-prompt set. Five findings emerge. (1) The static operand-count curriculum *collapses:* entropy rises to 11.69, importance weights to 8.75, and reward decays to zero by about step 60 with no recovery–hard stage boundaries combined with sparse rewards appear to push the policy out of distribution. (2) The SFT-estimated curricula avoid the collapse observed in the static curriculum across 100 steps, indicating the model-aware signal matches trainable difficulty better than operand count. (3) At the standard learning rate (1e-5), the SFT curriculum exhibits *peak-then-decay:* it peaks around step 70 and degrades by step 100 as harder and full-data prompts enter. (4) Recomputing difficulty over 20k prompts and lowering the learning rate to 5e-6 stabilizes training; the delayed 30/70 schedule reaches 78% pass@16 at step 80 (versus roughly 76% for the final baseline checkpoint). (5) Under higher-temperature sampling (temperature 0.8), the curriculum reaches 80% pass@16 against the baseline's 76%, suggesting broader exploration and greater robustness even where the two are tied at default temperature.

**Discussion, Limitations, and Conclusion** The contribution is diagnostic rather than a leaderboard result. The value lies in identifying *why* a naive curriculum fails and what mitigates it, not in maximizing a single number. Two limitations bound the claims. First, the comparisons are configuration-level rather than controlled single-variable ablations–schedule shape, learning rate, and difficulty-scoring set size co-vary–so I do not isolate each lever's individual effect. Second, the 50-prompt evaluation set means gaps of a few points fall within noise. The take-home message is that a model-aware difficulty signal paired with gentler optimization can prevent the catastrophic collapse that hand-crafted ordering induces under sparse rewards, turning a curriculum from a liability into a stabilizer. Promising next steps include KL regularization to bound drift, an adaptive curriculum that rescores difficulty online, early stopping to capture the peak before decay, and controlled ablations that vary one component at a time.

# SFT-Estimated Curriculum Learning for Rule-Based RLOO Fine-Tuning

**Vanessa Felix**
Department of Computer Science
Stanford University
vfelix@stanford.edu

## Abstract

Online reinforcement-learning fine-tuning of language models on sparse, verifiable-reward tasks suffers from a cold-start problem: early in training, the policy rarely produces a correct rollout, so most updates carry little learning signal. I investigate whether ordering prompts by difficulty, a curriculum, improves the stability and sample efficiency of RLOO fine-tuning, using the Countdown arithmetic task and Qwen2.5-0.5B. I compare a no-curriculum baseline against a static curriculum ordered by operand count and curricula ordered by an SFT-estimated difficulty signal, in which the supervised fine-tuned checkpoint itself scores each prompt by its empirical mean-shaped reward, a model-aware difficulty estimate rather than a hand-crafted heuristic. The difficulty signal, not the mere presence of a curriculum, appears decisive: the static operand-count curriculum collapses, with entropy and importance weights diverging and reward falling to zero, whereas the SFT-difficulty curricula avoid collapse and, with a lower learning rate and delayed stage transitions, train stably to 78% empirical pass@16–competitive with the $\sim 80\%$ baseline and more robust under higher-temperature sampling (80% versus 76%). These findings indicate that a model-aware difficulty signal paired with gentler optimization can turn a curriculum from a source of catastrophic instability into a stabilizer. The broader lesson is that curriculum design depends critically on how difficulty is defined, not merely on whether prompts are ordered.

## 1 Introduction

Reinforcement learning has become an important tool for improving the reasoning behavior of language models, especially when tasks can be evaluated with verifiable rewards rather than human preference labels DeepSeek-AI (2025). In mathematical and symbolic reasoning domains, a model can be rewarded automatically when its final answer satisfies a known constraint. This makes online RL attractive because it can, in principle, improve reasoning beyond supervised imitation. However, sparse verifiable rewards also create a difficult optimization problem: most sampled rollouts may receive little or no useful reward, especially early in training, when the policy has not yet learned to reliably produce valid solutions. As a result, training can waste substantial compute on uninformative samples and can become unstable when high-variance policy updates push the model away from useful behavior.

This project studies this problem in the setting of Countdown arithmetic, a constrained reasoning task in which the model must combine a given set of numbers using arithmetic operations to reach a target value. Countdown is a useful testbed for RL fine-tuning because correctness is automatically checkable, but successful rollouts require both valid formatting and exact arithmetic. A rollout that is fluent or plausible-looking may still receive low reward if the expression is invalid or does not equal

the target. This makes the task representative of a broader class of verifiable reasoning problems where reward is available, but sparse.

Even after supervised fine-tuning (SFT) provides a stronger starting point, sparse rewards can keep early updates inefficient, and high-variance updates on unproductive prompts risk destabilizing the policy before it bootstraps. This failure mode is not specific to one task. It appears wherever rewards are sparse and verifiable, including code generation, theorem proving, tool use, and multi-step mathematical reasoning, and it is most acute in resource-constrained settings, such as fine-tuning a small model on a single GPU under a fixed compute budget, where every wasted rollout matters.

A natural remedy is curriculum-based prompt ordering: present easier problems first so the policy meets prompts it can actually solve, harvesting dense early reward before facing harder ones. Recent curriculum methods for LLM reasoning use easy-to-hard schedules Parashar et al. (2025), self-evolving difficulty Chen et al. (2025), or teacher-generated stepping-stone problems Sundaram et al. (2026). These works motivate curriculum design but often require learned schedules, generated data, or large-scale training pipelines. They also leave open a question the RL-for-reasoning setting rarely answers cleanly: *by what measure of difficulty*? A hand-crafted proxy, such as the number of operands in an arithmetic problem, is cheap but may not reflect what a given model actually finds hard. An alternative is a model-aware signal that estimates difficulty from the policy's own competence, for example, the SFT checkpoint's empirical pass rate on each prompt.

I compare three prompt-ordering conditions. The first is a no-curriculum RLOO baseline, which samples prompts without explicit difficulty ordering. The second is a static curriculum based on operand count, where 3-operand problems are presented before 4-operand problems. The third is a family of SFT-estimated curricula, where the supervised fine-tuned checkpoint is used as a model-aware difficulty estimator. Each prompt is scored by empirical pass rate over sampled rollouts, and prompts are ordered from easier to harder according to this estimate. This approach requires no architecture changes and does not train a separate curriculum policy. Instead, it asks whether the model's own success distribution can provide a more useful curriculum signal than hand-crafted task metadata.

The project is guided by three research questions. First, does curriculum-based prompt ordering improve the stability and sample efficiency of sparse-reward RLOO fine-tuning relative to no curriculum? Second, does the *definition* of difficulty, a hand-crafted operand-count heuristic versus a model-aware, SFT-estimated signal, determine whether a curriculum helps or harms? Third, how do optimization choices such as learning rate and stage-transition timing interact with curriculum structure to govern stability? To answer these questions, I evaluate reward trajectories, empirical pass@16, response-level correctness, entropy, and importance-weight diagnostics across RLOO runs.

My initial hypothesis was that ordering prompts by difficulty would keep early reward dense and thereby improve stability and sample efficiency over no curriculum, with a model-aware signal outperforming a hand-crafted one. The results only partly bear this out, and the most informative findings run contrary to it. Under the main evaluation setting, the baseline matched or exceeded every curriculum variant, so curriculum ordering did not yield higher default-temperature pass@16. The positive results were narrower but instructive. The static operand-count curriculum *collapsed*. Entropy and importance weights rose sharply, and reward fell to zero, whereas the SFT-estimated curricula avoided this collapse. With a lower learning rate and delayed stage transitions, the model-aware curriculum recovered to within roughly one prompt of the baseline and showed modestly greater robustness under higher-temperature sampling. These results suggest that prompt ordering alone is insufficient. Curriculum behavior depends strongly on how difficulty is defined and how gently the policy is optimized.

The main contribution of this work is therefore diagnostic rather than leaderboard-oriented. The experiments show that curriculum design for sparse-reward RL fine-tuning depends critically on the definition of difficulty and on optimization stability. A naive hand-crafted curriculum can amplify instability, while a model-aware difficulty estimate paired with gentler optimization can act as a stabilizer. These findings suggest that future curriculum methods for language-model RL should treat difficulty estimation as a central design choice rather than a minor data-ordering detail.

## 2 Related Work

A growing body of work fine-tunes language models with reinforcement learning against automatically verifiable rewards rather than learned preference models, using the correctness of a final answer, a passing test suite, or a satisfied constraint as the training signal DeepSeek-AI (2025). This paradigm has produced strong reasoning gains, but it inherits a structural difficulty: when correct rollouts are rare at initialization, the reward is sparse, and online RL must obtain enough successes to bootstrap before it can improve. My work targets this sparse-reward regime directly, asking how the *ordering* of training prompts affects whether online RL stabilizes or diverges.

Several recent methods improve reasoning RL by controlling the difficulty or order of training problems. Easy-to-hard schedules present simpler problems first and report improved reasoning Parashar et al. (2025); self-evolving curricula adapt problem difficulty online as the policy improves Chen et al. (2025); and self-improvement approaches let a model generate its own stepping-stone problems to scaffold learning Sundaram et al. (2026). These works establish that difficulty-aware training can help, but they typically rely on additional machinery, such as a learned or adaptive scheduling policy, generated training data, or a large-scale pipeline that adds cost and complexity. By contrast, my approach is deliberately lightweight. It introduces no architecture changes, trains no separate curriculum policy, and generates no new data, instead reusing the SFT checkpoint the run already produces as its difficulty estimator.

The methods above motivate the importance of curriculum design, but they leave open a practical question that my project investigates in a small-scale RLOO setting: how should difficulty be estimated when rewards are sparse? Here, the number of operands in a Countdown problem, a hand-crafted structural proxy, is essentially free but does not necessarily align with what a given model actually finds hard. A model-aware estimate instead scores each prompt by the SFT checkpoint's empirical pass rate, grounding difficulty in the policy's own competence at initialization. Adaptive methods such as Adaptive Difficulty Curriculum Learning similarly estimate difficulty from the model's evolving state, but require an online scheduling loop that periodically re-estimates difficulty during training Zhang et al. (2025); I instead compute a fixed estimate once from the SFT checkpoint, which is simpler to add to a standard RLOO pipeline. To my knowledge, prior curriculum work for reasoning RL has not directly contrasted a hand-crafted structural signal against a same-model SFT-pass-rate signal within the same sparse-reward Countdown setting. This comparison, together with the training-stability diagnostics showing collapse of the structural curriculum, is the novel contribution of this work.

Another related direction uses language models to generate intermediate problems or stepping-stone curricula. Sundaram et al. study whether models stuck on sparse-reward hard problems can teach themselves by generating problems that bridge the gap between current capability and target difficulty Sundaram et al. (2026). This addresses a similar cold-start issue: when the initial success rate is too low, RL receives little useful signal. My project does not generate new data or train a teacher model. Instead, it asks a more constrained question: given a fixed Countdown dataset, can reordering existing prompts using an SFT-estimated difficulty signal improve RLOO stability? This distinction is important because prompt reordering is cheaper and easier to reproduce than synthetic data generation, but may be less powerful.

Overall, prior work suggests that curriculum learning can improve RL fine-tuning for reasoning, but also that the choice of curriculum signal and schedule is crucial. My project contributes a small-scale diagnostic study of this issue. Unlike methods that learn an online curriculum policy, generate new tasks, or rely on larger training pipelines, this extension modifies only prompt ordering in a standard RLOO setup. The comparison between operand-count ordering and SFT-estimated ordering investigates a practical design question: should curriculum difficulty come from task metadata or from the model's own success distribution? The results show that this choice matters: the static operand-count curriculum collapses, while SFT-estimated curricula avoid that collapse. Thus, the novelty of this work lies in using the SFT checkpoint as a lightweight difficulty estimator and in diagnosing curriculum failure modes under sparse-reward RLOO, rather than proposing a new RL objective or claiming a new state-of-the-art result.

# 3 Method

This project studies curriculum design for sparse-reward RLOO fine-tuning on the Countdown arithmetic task. The extension modifies the order in which prompts are presented during online RL while keeping the model and RLOO objective fixed. I compare three families of prompt ordering: a no-curriculum baseline, a static operand-count curriculum, and an SFT-estimated curriculum. The central question is whether curriculum difficulty should be defined by hand-crafted task metadata or by the model's own empirical reward distribution.

## 3.1 Task and Policy Setup

Each Countdown prompt $x$ specifies a target value and a multiset of input integers. A response $y$ is a generated arithmetic expression that must use the provided integers and evaluate to the target. The dataset is `asingh15/countdown_tasks_3to4`, containing roughly $490$k problems with three or four operands. A rule-based verifier assigns a *shaped* reward

$$R(x, y) \in \{0,\ 0.1,\ 1.0\},$$

equal to $0$ when the output contains no parseable `<answer>` span, $0.1$ (a format term) when the answer is well formed but the equation is invalid (does not use exactly the provided numbers) or does not reach the target, and $1.0$ when the equation is valid and evaluates to the target within a tolerance of $10^{-5}$. The reward is therefore not fully binary. The intermediate tier gives partial credit (0.1) for well-formed output, while full credit (1.0) requires an expression that exactly reaches the target. The *success* signal is nonetheless sparse early in training. Few rollouts reach the target, and when all samples for a prompt receive only the format term, the leave-one-out advantage (below) is near zero, so the update carries little learning signal. This is the cold-start regime the curriculum is intended to address.

All conditions start from the same supervised fine-tuned (SFT) checkpoint of the Qwen2.5-0.5B base model, denoted $\pi_{\theta_0}$, trained for 6 epochs at learning rate $5 \times 10^{-5}$. I initialize from SFT rather than the base model because prior work on Countdown reports that the $0.5$B base model fails to learn the task from RL alone Pan et al. (2025). Without some initial successes, there is no positive reward to learn from, so SFT's nonzero success rate is what lets RL bootstrap at all. The policy, reference model, tokenizer, and difficulty-scoring model all use this checkpoint. For practical reasons, three of the five runs reference it by local path, and two reference its Hugging Face Hub upload (`asingh15/qwen-sft-countdown-defaultproj`). I treat the two references as the same SFT checkpoint. Within the controlled static-vs-SFT curriculum comparison, the extension changes only the difficulty signal used to order prompts. Across the broader set of reported runs, later configurations also vary the stage schedule and optimization hyperparameters, so I treat baseline-vs-curriculum comparisons as configuration-level (Section 3.5).

## 3.2 RLOO Fine-Tuning

For each training prompt $x$, the policy samples a group of $K$ completions $y_1, \ldots, y_K \sim \pi_\theta(\cdot \mid x)$, each receiving a scalar reward $r_i = R(x, y_i)$. RLOO Ahmadian et al. (2024) estimates the advantage of each completion by comparing its reward to the mean reward of the others for the same prompt,

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

and updates the policy with the gradient estimate $\sum_i A_i \nabla_\theta \log \pi_\theta(y_i \mid x)$. Because the focus is curriculum design, I treat RLOO as a fixed optimization backbone so that the curriculum experiments isolate the effect of prompt ordering under the same reward.

Prompts are truncated to $512$ tokens and responses to $1024$ tokens, within a maximum model length of $2048$. Each run trains for $100$ RLOO steps with a constant learning-rate schedule, gradient clipping at $1.0$, and no warmup, checkpointing every $10$ steps.

The remaining optimization hyperparameters differ by experimental group, and I report them per condition rather than claiming a single shared configuration. The four curriculum runs (the static curriculum and the three SFT-estimated schedules), all in the same training sweep, share batch size $4$, group size $K = 2$, gradient accumulation $1$, entropy coefficient $0.01$, weight decay $0.01$, and

KL coefficient $0$ (no KL penalty to the SFT reference). The no-curriculum baseline was run in an earlier sweep with different defaults with batch size $128$, group size $K = 8$, gradient accumulation $128$, entropy coefficient $0.001$, weight decay $10^{-4}$, and KL coefficient $0.001$. I state this difference explicitly because it bears on which comparisons are controlled (Section 3.5).

Setting the KL coefficient to $0$ in the curriculum runs is a deliberate simplification: it removes one regularizer so that prompt ordering is the only intervention among those runs, but it also means nothing explicitly anchors the policy to its SFT initialization. I return to this in the Discussion in connection with the observed late-stage drift.

### 3.3 Prompt-Ordering Conditions

**No-curriculum baseline.** Prompts are sampled in random order with no difficulty structure, testing whether curriculum structure helps beyond ordinary RLOO fine-tuning from the SFT checkpoint.

**Static operand-count curriculum.** Difficulty is a hand-crafted structural proxy: the number of operands. Prompts with $3$ operands are treated as easier and presented before $4$-operand prompts. This requires no model evaluation but assumes operand count aligns with the model's actual probability of solving a prompt, which is an assumption the project tests directly.

**SFT-estimated curriculum.** Difficulty is model-aware. Before RL, I sample $M$ completions from the SFT checkpoint for each candidate prompt and record the empirical *mean shaped reward*,

$$\hat{s}_{\text{SFT}}(x) = \frac{1}{M} \sum_{m=1}^{M} R(x, y_m), \qquad y_m \sim \pi_{\theta_0}(\cdot \mid x),$$

and define the difficulty score as $d_{\text{SFT}}(x) = 1 - \hat{s}_{\text{SFT}}(x)$. Prompts are ordered from easiest to hardest, i.e. by ascending $d_{\text{SFT}}$ (equivalently, descending mean SFT reward), so problems on which the SFT model already earns higher reward are introduced earlier. Because the reward is the three-tier shaped signal, $\hat{s}_{\text{SFT}}$ is a mean reward rather than a binary pass rate. A reliably well-formed but wrong prompt scores near $0.1$, not $0$. The signal therefore mixes formatting competence with task success. This grounds difficulty in the model's reward under its own policy at initialization rather than in task metadata. The estimate is computed over a random sample of training prompts. I first scored $5$k prompts in early exploratory runs, then scaled to $20$k prompts for the final configuration. A key assumption is that this estimate is computed *once* before RL and held fixed throughout training. This keeps the method lightweight, but the ranking may drift out of alignment as the policy improves, a limitation I revisit in the Discussion.

### 3.4 Staged Curriculum Release

For the curriculum conditions, the ordered training set is exposed in three stages rather than all at once. Let $\mathcal{R}$ be the prompts sorted from easiest to hardest (by operand count or by $d_{\text{SFT}}$). At step $t$, the sampler draws from a prefix of $\mathcal{R}$,

$$\mathcal{P}_t = \mathcal{R}\big[\,1 : \lfloor f_t\,|\mathcal{R}|\rfloor\,\big],$$

where the stage fraction $f_t$ increases at fixed transition steps. I evaluate several schedules within the SFT-estimated family in addition to the static curriculum (Table 1 lists every run). The schedules vary in how much of the ranked data is exposed and how early transitions occur, and the final SFT configuration additionally lowers the learning rate from $1 \times 10^{-5}$ to $5 \times 10^{-6}$ after I observed late-stage degradation (a peak-then-decay pattern) at the higher rate. Here, slowing optimization reduces drift once harder, full-data prompts enter.

### 3.5 Scope of Comparisons

Because the runs differ in more than one factor, I am explicit about which comparisons are controlled. The *static* and *SFT (33/66)* runs are identical in every hyperparameter like batch size, group size, KL, entropy, weight decay, learning rate, stage fractions, and transition steps. They differ *only* in the difficulty signal (operand count vs. SFT mean reward). This pair is therefore a controlled, single-variable comparison, and it is the basis for attributing the collapse described in the Results to the difficulty signal rather than to the curriculum machinery.

Table 1: All five RLOO runs. The curriculum runs share batch size $4$, group size $2$, KL $0$, entropy $0.01$, and weight decay $0.01$; the baseline used batch $128$, group $8$, KL $0.001$, entropy $0.001$, weight decay $10^{-4}$. Difficulty scoring for the SFT runs used 20k prompts for the final configuration.

| Run | Difficulty signal | Stage fractions | Transition steps | LR |
|---|---|---|---|---|
| Baseline | none (random) | — | — | $1 \times 10^{-5}$ |
| Static | operand count | $0.33/0.66/1$ | $33/66$ | $1 \times 10^{-5}$ |
| SFT (33/66) | SFT mean reward | $0.33/0.66/1$ | $33/66$ | $1 \times 10^{-5}$ |
| SFT (soft) | SFT mean reward | $0.66/0.90/1$ | $20/50$ | $1 \times 10^{-5}$ |
| SFT (final) | SFT mean reward | $0.50/0.75/1$ | $30/70$ | $5 \times 10^{-6}$ |

**Algorithm 1** SFT-estimated difficulty scoring

**Require:** SFT policy $\pi_{\theta_0}$, prompt pool $\mathcal{D}$, sample size $N$, rollouts per prompt $M$, reward $R$

Sample $N$ prompts $\{x_1, \ldots, x_N\}$ from $\mathcal{D}$
**for** each prompt $x_j$ **do**
  Sample $M$ completions $y_1, \ldots, y_M \sim \pi_{\theta_0}(\cdot \mid x_j)$
  $\hat{s}_{\text{SFT}}(x_j) \leftarrow \frac{1}{M} \sum_{m=1}^{M} R(x_j, y_m)$ ▷ mean shaped reward
  $d_{\text{SFT}}(x_j) \leftarrow 1 - \hat{s}_{\text{SFT}}(x_j)$ ▷ difficulty
**end for**
Sort prompts by $d_{\text{SFT}}$ in ascending order (easiest first)
**return** ranked prompt list $\mathcal{R}$

The remaining comparisons are configuration-level rather than single-variable. The final SFT run differs from the static curriculum in difficulty signal, stage schedule, learning rate, and scoring-pool size simultaneously, and the no-curriculum baseline was trained in a separate sweep with different batch size, group size, KL coefficient, and entropy. I therefore treat baseline-vs-curriculum and best-config comparisons as configuration-level: they characterize the behavior of full training setups, not the marginal effect of a single knob.

### 3.6 Algorithms

Algorithm 1 computes the model-aware difficulty ranking; Algorithm 2 runs staged-curriculum RLOO given a ranking and the stage schedules. The baseline is recovered by setting $\mathcal{P} = \mathcal{R}$ (the full, unordered pool) at every step.

### 3.7 Evaluation

I evaluate on a held-out set of $50$ prompts, sampling $16$ rollouts per prompt and reporting *empirical pass@16* (the fraction of prompts with at least one fully correct response among the $16$ rollouts) and *response-level pass@1* (the fraction of all sampled responses that are fully correct). Both metrics threshold on full correctness ($R = 1.0$) and ignore the format term, so they measure task success rather than formatting. This distinguishes the *evaluation* criterion from the three-tier *training* reward. Evaluation uses temperature $0.6$ by default, and I additionally report results at temperature $0.8$ to probe robustness under higher-temperature sampling.

### 3.8 Novelty Relative to Prior Work

The extension is the use of the SFT checkpoint as a *fixed, model-aware difficulty estimator* for prompt ordering, evaluated head-to-head against a hand-crafted structural proxy under otherwise identical settings. Unlike SEC Chen et al. (2025), which learns an online curriculum policy as a non-stationary bandit, or ADCL Zhang et al. (2025), which re-estimates difficulty online during training, the estimator here is computed once and adds no online machinery to the RLOO loop. Unlike SOAR Sundaram et al. (2026), which generates new stepping-stone problems, I reorder existing prompts and synthesize no data. The contribution is therefore both a lightweight method and a controlled diagnostic comparison: holding all training hyperparameters fixed and varying only the difficulty signal (the static vs. SFT (33/66) pair), does difficulty defined by task metadata behave

**Algorithm 2** Staged-curriculum RLOO fine-tuning. In the reported curriculum runs, $\beta = 0.01$, $\lambda = 0$ (no KL term), and the importance-weight clamp is $c = 10$; the baseline uses $\beta = 0.001$ and $\lambda = 0.001$.

**Require:** SFT-init policy $\pi_{\theta_0}$, ranked prompts $\mathcal{R}$, fractions $\{f_1, f_2, f_3\}$, transition steps $\{t_1, t_2\}$, total steps $T$, group size $K$, reward $R$, entropy coef. $\beta$, KL coef. $\lambda$, IW clamp $c$

$\pi_\theta \leftarrow \pi_{\theta_0}$; reference $\pi_{\text{ref}} \leftarrow \pi_{\theta_0}$ ▷ $\pi_{\text{ref}}$ loaded only if $\lambda > 0$
**for** step $= 0$ to $T - 1$ **do**
  $f \leftarrow f_1$ if step $< t_1$; $f_2$ if $t_1 \le$ step $< t_2$; else $f_3$
  $\mathcal{P} \leftarrow$ easiest $f$ fraction of $\mathcal{R}$
  Sample a batch of prompts from $\mathcal{P}$
  **for** each prompt $x$ in the batch **do**
    **for** $i = 1$ to $K$ **do**
      Sample $y_i \sim \pi_{\text{samp}}(\cdot \mid x)$, storing behavior log-prob $b_i = \log \pi_{\text{samp}}(y_i \mid x)$
      $r_i \leftarrow R(x, y_i)$
    **end for**
    $A_i \leftarrow r_i - \frac{1}{K-1} \sum_{j \neq i} r_j$ for $i = 1, \ldots, K$ ▷ leave-one-out baseline
    **for** $i = 1$ to $K$ **do**
      $\ell_i \leftarrow \sum_t \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$ ▷ summed response log-prob
      $w_i \leftarrow \min\big(\exp(\ell_i - b_i),\ c\big)$ ▷ clamped importance weight
    **end for**
  **end for**
  $\mathcal{L}_{\text{PG}} \leftarrow -\,\text{mean}_i\big[\, w_i\, A_i\, \ell_i / |y_i| \,\big]$ ▷ response-length normalized
  $\mathcal{L}_{\text{ent}} \leftarrow -\beta \cdot \text{mean}_i\, \bar{H}_i$ ▷ $\bar{H}_i$: mean per-token entropy over response tokens
  $\mathcal{L}_{\text{KL}} \leftarrow \lambda \cdot \text{mean}_i\big[(\ell_i - \ell_i^{\text{ref}})/|y_i|\big]$ ▷ computed only if $\lambda > 0$
  Apply AdamW update to minimize $\mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{KL}}$ with gradient clipping
**end for**
**return** fine-tuned policy $\pi_\theta$

differently from difficulty defined by the model's own reward distribution? As the Results show, it does. The structural curriculum collapses while the model-aware one does not.

# 4 Experimental Setup

## 4.1 Dataset and Task

I evaluate curriculum RLOO on the Countdown arithmetic task using `asingh15/countdown_tasks_3to4`. Each prompt provides a target number and a multiset of three or four integers, and the model must generate an arithmetic expression that uses exactly the provided numbers and evaluates to the target. The task is well-suited for verifiable-reward RL because correctness can be checked automatically by a rule-based verifier, but it is also sparse in the success signal. Full reward is obtained only when the generated expression is valid and matches the target exactly. As described in the Method, the training reward is shaped with three possible values, $0$, $0.1$, and $1.0$, where $0.1$ rewards a correctly formatted but invalid or incorrect answer. Thus, the reward is not purely binary, but exact task success remains sparse early in training. The training split contains roughly $490$k prompts. All runs initialize from an SFT-trained Qwen2.5-0.5B checkpoint, which provides a nonzero initial success rate before online RL.

## 4.2 Compared Runs and Baselines

I compare five RLOO configurations. The main baseline is no-curriculum RLOO, where prompts are sampled in random order from the training set. This baseline tests whether curriculum ordering improves over ordinary online RLOO fine-tuning from the same SFT initialization. I also include a static curriculum baseline ordered by operand count, which tests a simple hand-crafted difficulty proxy: three-operand problems are treated as easier than four-operand problems. This is a useful baseline because it represents the cheapest possible curriculum signal, requiring no model sampling or learned scheduler.

Table 2: Key optimization and sampling hyperparameters.

| Hyperparameter | Curriculum runs | No-curriculum baseline |
|---|---|---|
| Training steps | 100 | 100 |
| Checkpoint interval | 10 steps | 10 steps |
| Training temperature | 1.0 | 1.0 |
| Top-$p$ / top-$k$ / min-$p$ | $1.0/-1/0$ | $1.0/-1/0$ |
| Max response length | 1024 | 1024 |
| Max model length | 2048 | 2048 |
| Batch size | 4 | 128 |
| Group size $K$ | 2 | 8 |
| Gradient accumulation | 1 | 128 |
| Learning rate | $1 \times 10^{-5}$; final delayed SFT uses $5 \times 10^{-6}$ | $1 \times 10^{-5}$ |
| Entropy coefficient | 0.01 | 0.001 |
| KL coefficient | 0 | 0.001 |
| Weight decay | 0.01 | $10^{-4}$ |
| Gradient clipping | 1.0 | 1.0 |

The proposed curriculum uses the SFT checkpoint itself to estimate prompt difficulty. For each scored prompt, I sample $M = 2$ responses from the SFT model and compute the mean shaped reward. Prompts with higher mean SFT reward are treated as easier and introduced earlier. Because $M = 2$ and the reward is three-tiered, the per-prompt mean reward takes only a small number of values. The resulting ranking is therefore coarse and should be understood as a noisy, lightweight difficulty estimate rather than a precise ordering.

In implementation, difficulty scores are computed for a sampled subset of training prompts. The curriculum dataset then sorts the full training split by these scores, placing prompts without an assigned score after scored prompts. Stage fractions are applied to this full sorted split. Thus, the active pool is a prefix of a partially ordered full training set, not necessarily a perfectly ranked easy-to-hard ordering of all 490k prompts. For the final delayed SFT configuration, the logged active-pool fractions follow the intended schedule: $0.50$, $0.75$, and $1.0$ at transition steps 30 and 70.

I evaluate three SFT-estimated curriculum schedules: an initial 33/66 schedule, a softer 20/50 schedule, and a final delayed 30/70 schedule with a lower learning rate. The static and SFT 33/66 runs are the controlled comparison. They share the same training hyperparameters and stage schedule, differing only in the difficulty signal. The remaining comparisons are configuration-level comparisons because schedule, learning rate, or scoring-pool size also change.

### 4.3 Training Details and Hyperparameters

All runs train for 100 RLOO steps and save checkpoints every 10 steps. Prompts are truncated to 512 tokens. The optimizer is AdamW with a constant learning-rate schedule, no warmup, and gradient clipping at 1.0.

The curriculum runs use batch size 4, group size $K = 2$, gradient accumulation 1, entropy coefficient $0.01$, KL coefficient 0, and weight decay $0.01$. The static, SFT 33/66, and soft SFT runs use learning rate $1 \times 10^{-5}$, while the final delayed SFT run lowers the learning rate to $5 \times 10^{-6}$. The no-curriculum baseline was run in an earlier sweep with batch size 128, group size $K = 8$, gradient accumulation 128, entropy coefficient $0.001$, KL coefficient $0.001$, and weight decay $10^{-4}$. Because the baseline and curriculum runs are not fully hyperparameter-matched, I treat baseline-vs-curriculum comparisons as comparisons between full training configurations rather than controlled single-variable ablations.

For SFT-based difficulty scoring, early exploratory runs used a smaller scored subset, while the final configuration scores 20k randomly sampled training prompts using the SFT checkpoint and $M = 2$ rollouts per prompt. The resulting ranking is computed once before RL training and held fixed throughout the run. Static curriculum scoring uses operand count directly and therefore requires no sampling pass.

Table 3: Default-temperature evaluation on 50 held-out prompts. Empirical pass@16 is the fraction of prompts for which at least one of 16 sampled responses is fully correct. All runs are reported at step 100 except the delayed SFT curriculum, which is reported at its best checkpoint, step 80. Confidence intervals are Wilson 95% intervals over prompts.

| Run | Checkpoint | Solved / 50 | Empirical pass@16 |
|---|---|---|---|
| No-curriculum baseline | step 100 | 38/50 | 76% [62.6, 85.7] |
| Delayed SFT curriculum | step 80 | 39/50 | 78% [64.8, 87.2] |
| Soft SFT curriculum | step 100 | 14/50 | 28% [17.5, 41.7] |
| SFT curriculum 33/66 | step 100 | 16/50 | 32% [20.8, 45.8] |
| Static curriculum | step 100 | 0/50 | 0% [0.0, 7.1] |

### 4.4 Evaluation Metrics

I evaluate each checkpoint on a held-out set of 50 prompts from the test split. For each prompt, I sample 16 responses and compute two task-success metrics. The primary metric is *empirical pass@16*: the fraction of prompts for which at least one of the 16 sampled responses is fully correct, meaning it receives reward 1.0. This metric is appropriate for Countdown because many uses of sampled reasoning models care whether the model can produce at least one valid solution under repeated sampling. I also report *response-level pass@1*, the fraction of all sampled responses that are fully correct. This metric measures the average quality of individual samples and is more sensitive to whether the policy assigns high probability to correct solutions.

Both evaluation metrics threshold on exact correctness and ignore the 0.1 format reward, so they measure task success rather than formatting compliance. Evaluation uses temperature 0.6 by default with 16 responses per prompt. I also evaluate selected checkpoints at temperature 0.8 to test whether the curriculum policy remains robust under higher-temperature sampling.

## 5 Results

I evaluate every checkpoint on the same held-out set of 50 prompts, sampling 16 responses per prompt at temperature 0.6 unless otherwise noted. I report *empirical pass@16*, the fraction of prompts for which at least one sampled response is fully correct, together with *response-level pass@1*, the fraction of all sampled responses that are fully correct. Confidence intervals are Wilson 95% intervals over the 50 held-out prompts and should be interpreted as finite-evaluation-set uncertainty, not as variance across random seeds.

The main result is mixed. The best curriculum configuration slightly exceeds the final no-curriculum baseline in empirical pass@16, but only by one prompt on a 50-prompt evaluation set, so the difference is not statistically meaningful. At the same time, the curriculum experiments reveal an important stability effect: a hand-crafted operand-count curriculum collapses completely, while SFT-estimated curricula avoid that zero-reward collapse and, with a delayed schedule and lower learning rate, recover to baseline-level pass@16.

### 5.1 Quantitative Evaluation

Table 3 summarizes the main default-temperature comparison. To keep checkpoints consistent, all runs are reported at step 100 except the delayed SFT curriculum, which is reported at its best checkpoint, step 80. The no-curriculum baseline solves 38/50 prompts (76% empirical pass@16), while the delayed SFT curriculum solves 39/50 prompts (78% empirical pass@16). This one-prompt difference is not meaningful on a 50-prompt evaluation set because the Wilson confidence intervals overlap substantially. Thus, the strongest curriculum result is best interpreted as competitive with, but not decisively better than, the baseline.

The table shows three patterns. First, ordinary RLOO from the SFT checkpoint is a strong baseline, and the best curriculum does not clearly exceed it. Second, curriculum ordering by itself is not sufficient because the static operand-count curriculum collapses to 0% empirical pass@16. Third, the SFT-estimated curricula are consistently better than the static curriculum, suggesting that model-aware reward estimates are a safer difficulty signal than operand count alone.

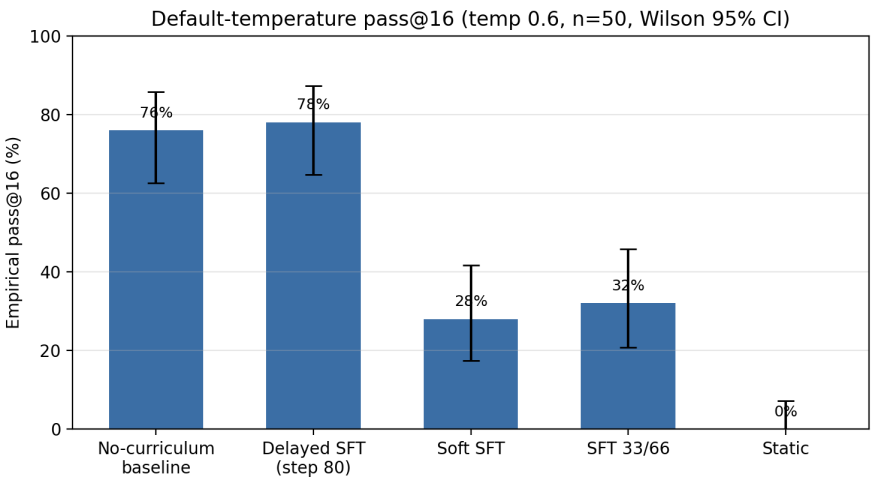


Figure 1: Default-temperature empirical pass@16. The delayed SFT curriculum reaches baseline-level performance, while the static operand-count curriculum collapses.

Table 4: Response-level reliability at representative checkpoints. Pass@1 is the fraction of all sampled responses that are fully correct.

| Run | Response-level pass@1 |
|---|---|
| SFT initialization | $29.0\%$ |
| No-curriculum baseline | $59.0\%$ |
| Delayed SFT curriculum | $32.6\%$ |
| SFT curriculum $33/66$ | $3.6\%$ |
| Soft SFT curriculum | $3.6\%$ |
| Static curriculum | $0.0\%$ |

The baseline's behavior is also informative when pass@1 is considered. Compared with the SFT initialization, the no-curriculum baseline substantially improves response-level reliability, where we see the fraction of individual sampled responses that are correct rises from $29.0\%$ to $59.0\%$. In contrast, the delayed SFT curriculum recovers pass@16 but reaches only $32.6\%$ response-level pass@1. This means the curriculum-trained model can still produce correct solutions under repeated sampling, but the no-curriculum baseline assigns much higher probability to correct responses on a single draw.

I also evaluate the baseline and best curriculum at temperature $0.8$ to test robustness under higher-temperature sampling. In this setting, the delayed SFT curriculum solves $40/50$ prompts ($80\%$), compared with $38/50$ prompts ($76\%$) for the baseline. This gap is again small and should not be overread, but it suggests that the curriculum-trained policy may preserve useful solution diversity under broader sampling.

### 5.2 Training Dynamics

Final pass@16 hides the main failure mode. Figure 3 shows the mean training reward over RLOO updates. The no-curriculum baseline improves steadily over training. The static curriculum initially receives some reward but collapses to near zero by roughly step $60$ and does not recover. This suggests that the operand-count stage transition exposes the policy to prompts that are poorly matched to its current capabilities, causing most sampled rollouts to lose useful reward contrast.

The SFT-estimated curricula behave differently. The initial SFT $33/66$ schedule does not collapse to zero, but it underperforms. The soft SFT schedule improves substantially earlier in training and peaks around step 70, but degrades by step $100$ as harder and full-data prompts enter. The final delayed SFT run, which uses stage transitions at steps $30$ and $70$ and a lower learning rate of $5 \times 10^{-6}$, stabilizes this behavior and reaches its best held-out result at step $80$.

Table 5: Higher-temperature evaluation at temperature $0.8$.

| Run | Solved / 50 | Empirical pass@16 |
|---|---|---|
| No-curriculum baseline | $38/50$ | $76\%$ $[62.6, 85.7]$ |
| Delayed SFT curriculum | $40/50$ | $80\%$ $[67.0, 88.8]$ |

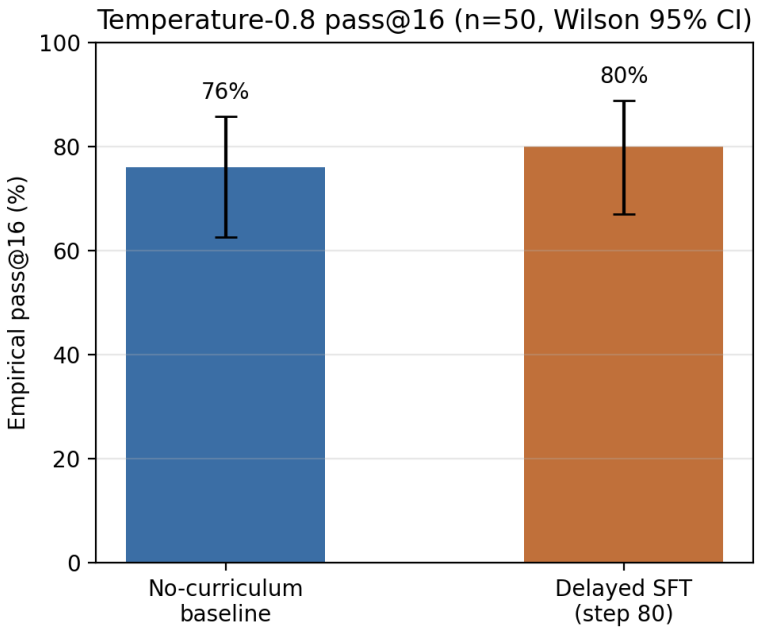


Figure 2: At temperature $0.8$, the delayed SFT curriculum slightly exceeds the baseline in empirical pass@16. Because the evaluation set contains only $50$ prompts, this is a robustness signal rather than a statistically decisive win.

These trajectories show why the project is best interpreted as a stability study, not a simple leaderboard result. At the default evaluation temperature, the best curriculum and baseline are statistically indistinguishable in empirical pass@16, while the baseline is much stronger in response-level pass@1. However, the static-vs-SFT comparison shows that the difficulty signal qualitatively changes training behavior, where we see task-metadata difficulty induces collapse, while model-aware difficulty avoids it.

### 5.3 Diagnostic Analysis of Static-Curriculum Collapse

The static curriculum failure is visible in both reward and optimization diagnostics. In the collapsed static run, mean reward falls to zero by roughly step $60$, while entropy rises to approximately $11.69$ and the mean importance weight rises to approximately $8.75$. Since importance weights are clamped at $10$, a mean value near $8.75$ indicates that many updates are occurring close to the clipping limit. This suggests that the policy distribution used for updates has drifted far from the distribution that generated the sampled rollouts, making updates high-variance and unstable.

This failure mode is consistent with sparse-reward cold start. In successful rollouts, the model must preserve the required answer format, use exactly the provided numbers, and construct an expression that reaches the target. Once the policy enters a stage where most completions are invalid, malformed, or incorrect, the shaped reward provides little useful distinction among completions. If all completions for a prompt receive the same reward, the leave-one-out advantage is near zero. If they receive noisy small differences, the update can become dominated by variance rather than useful learning signal. Thus, a curriculum can be actively harmful when its difficulty signal is not aligned with the model's actual competence.

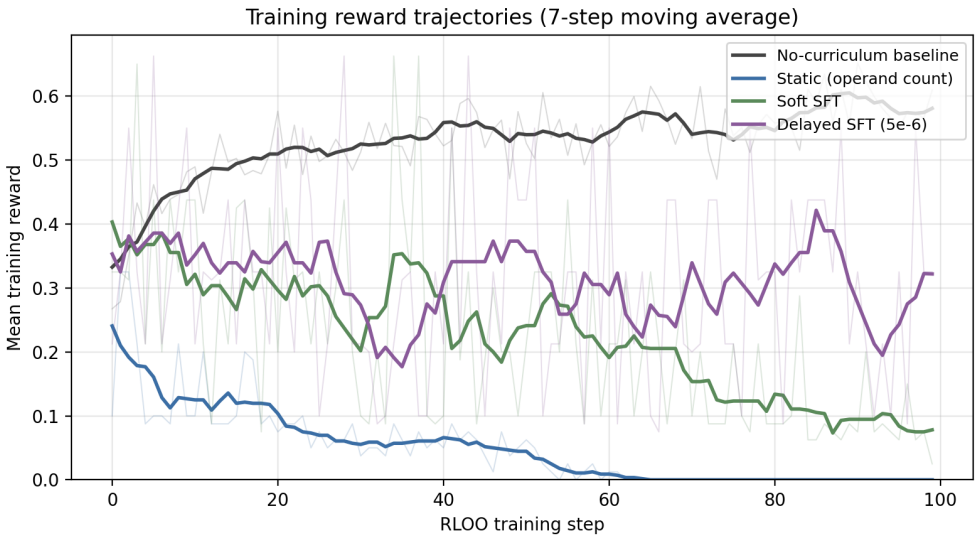


Figure 3: Training reward trajectories. The static operand-count curriculum collapses to zero reward, while SFT-estimated curricula avoid catastrophic collapse. The delayed SFT schedule with lower learning rate is the most stable curriculum configuration.

### 5.4 Qualitative Rollout Analysis

The aggregate metrics are reflected in the model's generations. On a prompt with target $98$ and numbers $\{44, 19, 35\}$, the delayed SFT curriculum produces a short, correct solution:

```
<think> Let me try to find a path to 98 ...  44 + 19 = 63; 63 + 35 =
98!  I found it ...</think> <answer> 44 + 19 + 35 </answer>
```

By contrast, the collapsed static run at step $100$ produces long, non-committal reasoning that does not reach a valid answer:

```
<think> Let me try to find useful subgoals first ...  If we could
somehow combine numbers close to target somehow ...  But I don't see
any clear paths.  Working backwards might be tricky too ...
```

Other collapsed generations show repetitive behavior, including repeated tokens until the length limit. The inspected static outputs typically remain well-formed enough to receive the $0.1$ format reward, but they no longer solve the arithmetic problem. This explains why the format-shaping term is not enough to rescue training because it preserves parseable output, but it does not provide a strong gradient toward correctness once the policy stops finding exact solutions.

The delayed SFT curriculum avoids this collapse and stays closer to the productive behavior of the SFT initialization. However, the qualitative examples also support the quantitative conclusion. We see that the delayed curriculum stabilizes training but does not clearly improve beyond the no-curriculum baseline. The baseline achieves similar empirical pass@16 and a much stronger response-level pass@1 without requiring a curriculum.

### 5.5 What Explains the Difference Between Curriculum Runs?

The controlled static-vs-SFT $33/66$ comparison is suggestive but not perfectly isolated. These runs share the same batch size, group size, learning rate, entropy coefficient, KL coefficient, weight decay, stage fractions, and transition steps, differing only in the difficulty signal. Static difficulty collapses to $0\%$ pass@16, while SFT-estimated difficulty reaches $32\%$ and avoids zero-reward collapse. This supports the hypothesis that model-aware difficulty is safer than operand count.

However, the interpretation should be cautious. The SFT difficulty estimate is coarse because it uses only $M = 2$ rollouts per scored prompt and a three-tier reward. In addition, only a subset of prompts

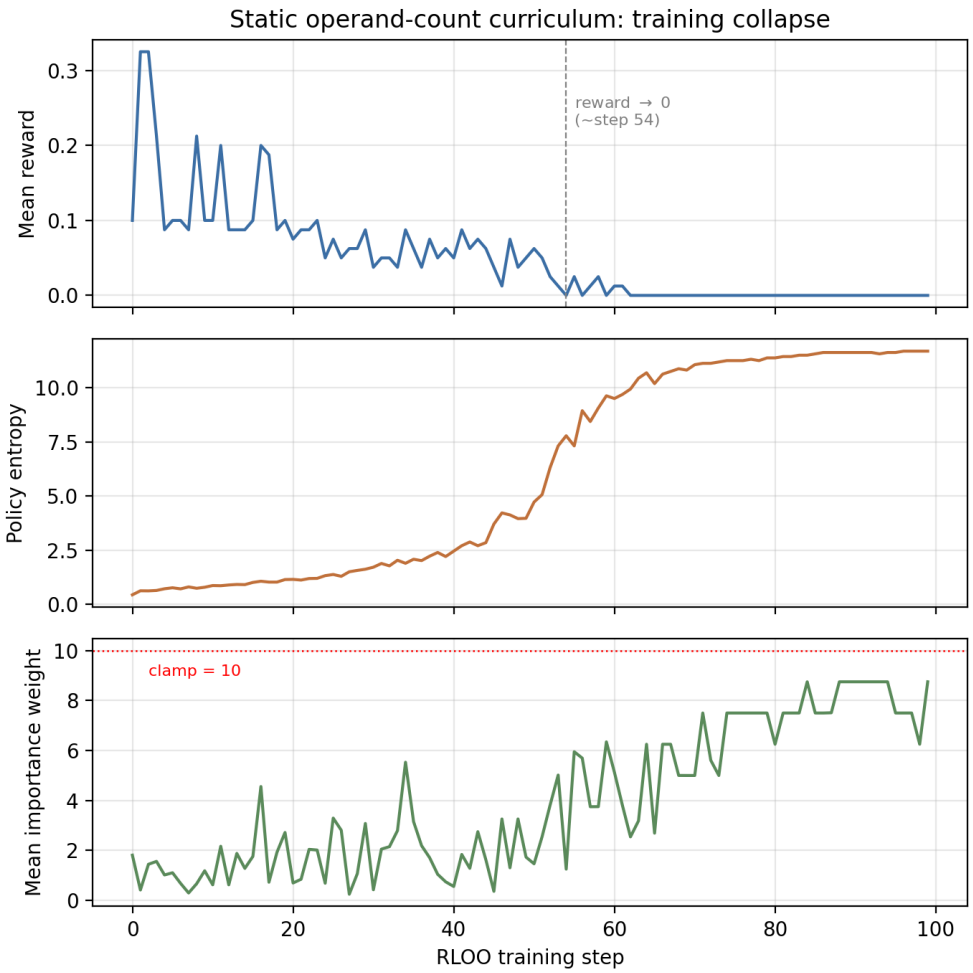


Figure 4: Static-curriculum diagnostics. The reward collapse coincides with high entropy and large importance weights, suggesting unstable policy updates after the curriculum enters harder stages.

is explicitly scored. Unscored prompts are placed after scored prompts in the full sorted training split, and stage fractions unlock prefixes of that partially ordered split. Therefore, the curriculum is a lightweight partial ordering rather than a precise global difficulty ranking over all 490k prompts. This helps explain why the SFT-estimated curriculum improves stability relative to the static curriculum but does not clearly outperform the no-curriculum baseline.

The final delayed run changes multiple factors at once, like learning rate, stage schedule, and scoring-pool size. Therefore, its improvement over earlier SFT curricula is configuration-level rather than a clean ablation. The pattern is nevertheless informative, where we see the higher-learning-rate SFT curricula show underperformance or late-stage decay, while the delayed lower-learning-rate run is more stable. This suggests that optimization step size and stage timing are important for preventing curriculum-induced drift, but additional controlled ablations would be needed to isolate their individual effects.

### 5.6 Summary of Findings

The results support four conclusions. First, curriculum ordering by itself is not sufficient because we saw the static operand-count curriculum collapse completely. Second, the difficulty signal matters because under the same $33/66$ schedule and optimization settings, replacing operand count with SFT-estimated mean reward avoids the zero-reward collapse. Third, schedule and optimization matter.

SFT curricula at learning rate $1 \times 10^{-5}$ either underperform or decay late in training, while the delayed lower-learning-rate configuration is more stable. Fourth, the best curriculum run is competitive but not decisively better than the no-curriculum baseline. It reaches $78\%$ empirical pass@16 at default temperature versus $76\%$ for the baseline, and $80\%$ versus $76\%$ at temperature $0.8$. Thus, the main contribution is not a new state-of-the-art score, but a diagnostic result where model-aware difficulty can prevent a curriculum-induced collapse that occurs with a hand-crafted structural difficulty signal.

## 6 Discussion

**Limitations.** The main limitation is that the curriculum signal was diluted. Difficulty scores were computed for only $\sim$20k prompts, while the first-stage active pool already contained $\sim$245k prompts. The unscored prompts filled the rest of the sorted split, so the intended easy-to-hard ordering only partially shaped each sampled batch. The static vs SFT comparison is therefore suggestive rather than a clean isolation of the difficulty signal. The SFT difficulty estimate is also coarse because it uses only $M = 2$ rollouts per prompt and a three-tier reward. Finally, all results use a single seed and a $50$-prompt evaluation set, so one or two-prompt differences are within uncertainty, so baseline vs curriculum comparisons are also configuration-level because batch size, group size, and regularization differ.

**Difficulties encountered.** The central difficulty was instability under sparse reward without a KL anchor ($\lambda = 0$). The static curriculum collapsed where reward fell to zero by roughly step $60$, while entropy and importance weights rose toward the clamp, indicating policy drift with little signal to recover. Reaching a stable curriculum run required iterating on the scoring pool ($5$k to $20$k), stage schedule, and learning rate, where the delayed $5 \times 10^{-6}$ configuration was the only curriculum run that remained competitive through $100$ steps. Diagnosing the issue was itself nontrivial because the curriculum appeared to run correctly, but the active-pool logs showed that the difficulty ordering was weaker than intended.

**Broader impacts.** This is a small-scale diagnostic study on a verifiable arithmetic task, so direct societal risks are limited. The broader lesson is cautionary: a curriculum whose difficulty signal is misaligned with the model's actual competence can be actively harmful rather than merely unhelpful. As curriculum and data-ordering methods are used for reasoning models, they should be validated not only by final accuracy, but also by whether they actually shape the training distribution and maintain stable reward, entropy, and importance-weight behavior.

## 7 Conclusion

I studied whether curriculum-based prompt ordering improves RLOO fine-tuning of a small language model on Countdown arithmetic, comparing a no-curriculum baseline, a static operand-count curriculum, and SFT-estimated difficulty curricula. The key takeaway is that curriculum ordering did not decisively beat the baseline, and the project is most useful as a stability diagnosis. The static curriculum collapsed completely to $0\%$ pass@16, SFT-estimated curricula avoided that zero-reward collapse, and only the delayed lower learning rate configuration recovered baseline level pass@16, though not the baseline's response-level reliability. The significance is cautionary rather than leaderboard-driven: under sparse reward without a KL anchor, a difficulty signal misaligned with the model's competence can actively destabilize training, so a plausible-looking curriculum is not safe by default. This conclusion is tempered by the fact that the difficulty ordering was diluted by unscored prompts, so the difficulty signal's effect could not be cleanly isolated.

Future work follows directly from these limitations. The clearest next step is to score the full training pool, or restrict the active pool to scored prompts, so the curriculum ordering genuinely shapes each batch, then re-run the controlled static-vs-SFT comparison. Beyond that, matched ablations over learning rate, KL coefficient, and scoring-pool size would separate optimization factors from the difficulty signal, and adaptive difficulty re-estimated during training may better track the model's moving competence.

## 8 Team Contributions

- **Vanessa Felix:** entire project

**Changes from Proposal** The proposal framed the extension as testing whether easy-to-hard curricula improve RLOO on Countdown. The final project broadened this into a stability diagnosis after the static operand-count curriculum collapsed. I added static-vs-SFT difficulty comparisons, response-level pass@1, Wilson confidence intervals, higher-temperature evaluation, and entropy/importance-weight diagnostics. A key implementation finding was that the intended curriculum signal was diluted where only $\sim$20k prompts were scored, while the first active pool already contained $\sim$245k prompts, so the difficulty ordering only partially shaped sampled batches. The resulting conclusion is more cautious than the proposal: curriculum ordering did not decisively beat the baseline, but model-aware difficulty helped avoid a curriculum-induced collapse.

## References


Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12248–12267.

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).

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

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. `https://github.com/Jiayi-Pan/TinyZero`. Accessed: 2025-01-24.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. 2025. Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning. *arXiv preprint arXiv:2506.06632* (2025).

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).

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Qianchun Lu. 2025. Learning Like Humans: Advancing LLM Reasoning Capabilities via Adaptive Difficulty Curriculum Learning and Expert-Guided Self-Reformulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6619–6633.


## A AI Tools Disclosure

ChatGPT was used for boilerplate code, debugging minor code issues, LaTeX/report formatting, and plotting-script assistance
Note: All key RL components were implemented independently without the assistance of AI