Extended Abstract

Motivation Reinforcement learning with verifiable rewards (RLVR) has become a standard tool for improving the reasoning ability of language models, but it is notoriously sample-inefficient. In online policy-gradient methods such as REINFORCE Leave-One-Out (RLOO), prompts are drawn uniformly at random regardless of difficulty. Two pathologies follow directly from this choice. Prompts that the policy already solves yield near-zero advantage and therefore contribute almost no learning signal, while prompts that the policy never solves yield no successful rollout at all and therefore no gradient. On the Countdown arithmetic-reasoning task, we observe that the supervised fine-tuned (SFT) initialization solves fewer than 5% of rollouts on the majority of prompts, and that vanilla RLOO begins training with more than 70% of rollouts earning zero reward. The central question of this work is whether *ordering* the prompts a model sees, and in particular adapting that order to the model's own measured competence, recovers some of this wasted compute and improves downstream accuracy.

Method We propose an *adaptive, success-rate-gated curriculum* for RLOO. We first score every training prompt by sampling $K{=}16$ rollouts from the frozen SFT checkpoint and taking the mean verifier reward $\bar{r}(x)$ as a difficulty proxy. Prompts are bucketed into Easy/Medium/Hard tiers using either a balanced percentile split (our primary scheme) or fixed reward thresholds (an ablation). Training begins on the Easy tier only; a controller monitors the rolling pass rate on a held-out validation slice of the active tier and *unlocks* the next tier — adding its prompts to the sampling pool — once that pass rate clears a tier-specific threshold. The novel contribution is that the controller is gated on the policy's own real-time success rate using only the verifier signal already present in the pipeline, rather than on a predefined offline schedule (Parashar et al., 2025), an auxiliary bandit policy (Chen et al., 2025), or a fixed-interval difficulty re-estimation (Zhang et al., 2025). To our knowledge this is the first study of how adaptive curriculum pacing interacts specifically with RLOO's leave-one-out baseline.

Implementation All six experiments (vanilla RLOO, fixed percentile curriculum, sorted fixed curriculum, adaptive percentile curriculum, adaptive threshold curriculum, and adaptive relative-improvement curriculum) train on an identical fixed 18,432-prompt subset for 120 steps at batch size 128, with zero overlap with the 50-prompt test set.

Results The adaptive percentile curriculum is best across every metric, reaching pass@1 $= 0.660$ and pass@16 $= 0.840$, gains of $+2.9$ and $+2.0$ points over vanilla RLOO. Neither the fixed curriculum nor the sorted fixed curriculum improve over vanilla, indicating that ordering alone is not responsible for the gain; performance gating is the active ingredient. The balanced percentile split also beats the imbalanced threshold split at every k. A plateau-based relative-improvement gating variant is competitive but slightly underperforms the absolute-threshold gating. Curriculum experiments cut the fraction of zero-reward rollouts sharply early in training and collect more reward per step throughout.

Discussion Adaptive curricula offer a small but consistent improvement on Countdown at no extra model or data cost, and meaningfully reduce wasted gradient computation. The result is specific to a single task and a single base model, and our hand-tuned gating thresholds were low enough that gating effectively collapsed to a fixed minimum-window schedule, suggesting the headroom from genuinely adaptive pacing is larger than what we measured.

Conclusion How a curriculum decides to advance matters more than the mere fact of going easy-to-hard: performance gating, not ordering, is what improves RLOO here. Online difficulty re-estimation and continuous (rather than three-bucket) frontier-aware sampling are the most promising next steps.

Adaptive Curriculum Learning for RL-Based Arithmetic Reasoning

Yuchi Hsu
Electrical Engineering
Stanford University
yuchihsu@stanford.edu

Ryan He
Computer Science
Stanford University
ryanhe@stanford.edu

Abstract

We study curriculum learning as an extension to RLOO fine-tuning on the Countdown arithmetic-reasoning task. Uniform prompt sampling wastes most of the gradient budget on prompts that are either already solved (zero advantage) or never solved (zero signal). We introduce an adaptive, success-rate-gated curriculum that estimates per-prompt difficulty once from the SFT model, buckets prompts into Easy/Medium/Hard tiers, and unlocks harder tiers only when the policy's measured pass rate on the current tier crosses a threshold. Against vanilla RLOO, two fixed curricula (discrete tiers and a smooth difficulty ramp), an imbalanced threshold-based ablation, and a plateau-based relative-improvement variant, the adaptive percentile curriculum is best at every k, improving pass@1 by 2.9 points and pass@16 by 2.0 points while sharply reducing the fraction of zero-reward rollouts. A controlled comparison shows the gain comes from performance gating rather than from ordering alone.

1 Introduction

Reinforcement learning with verifiable rewards now underpins much of the recent progress in language-model reasoning (DeepSeek-AI, 2025). The recipe is attractively simple: a policy samples candidate solutions, an automatic verifier scores them, and a policy gradient pushes probability mass toward the rewarded completions. RLOO (**?**) is a clean instance of this family, using a leave-one-out average over a group of samples as a variance-reducing baseline for REINFORCE.

A subtle inefficiency hides inside this loop. RLOO's advantage for a sample is its reward minus the mean reward of its sibling samples on the same prompt. When all k samples for a prompt succeed, the advantages are all near zero and the prompt contributes almost nothing to the update. When all k samples fail, which is the common case for hard prompts early in training, every reward is zero, the advantages are again zero, and there is no learning signal. Only prompts on the policy's competence frontier, where some samples succeed and others fail, produce informative gradients. Under uniform sampling, the fraction of such prompts is small and shrinks as easy prompts are mastered, so a large share of every batch is wasted.

This observation motivates a curriculum: deliberately concentrate early training on prompts the model can occasionally solve, then expand to harder prompts as competence grows. The classical curriculum-learning hypothesis (Bengio et al., 2009) predicts that such easy-to-hard ordering can accelerate convergence and improve generalization. We ask two concrete research questions in the RLOO setting:

1. **Does the order in which the model sees prompts matter** for final Countdown accuracy and sample efficiency?

2. **Does adaptive, performance-gated pacing beat a fixed schedule**, or is any benefit attributable to ordering alone?

Our hypothesis is that adaptivity — timing tier transitions to the policy's own measured success rate — is necessary, because the right moment to introduce harder prompts depends on how fast the particular policy is learning, which a fixed schedule cannot know in advance.

2 Related Work

Curriculum learning. Bengio et al. (2009) formalized the idea that ordering training examples from easy to hard can speed convergence and improve generalization, and self-paced learning (Kumar et al., 2010) let the model itself decide which examples are "easy" enough to include. In the RL setting, Narvekar et al. (2020) survey curricula over tasks, and Portelas et al. (2020) review automatic curriculum generation for deep RL. These works predate LLM fine-tuning and do not address the discrete token action space or the sparse, verifier-based rewards that characterize RL post-training of language models.

Curricula for LLM RL fine-tuning. Closest to us, Parashar et al. (2025) propose E2H Reasoner, a probabilistic scheduler that gradually shifts focus from easy to hard tasks and evaluates on Countdown among other benchmarks; however, its schedule is fixed offline rather than gated on the training model's real-time success rate. Chen et al. (2025) cast curriculum selection as a non-stationary multi-armed bandit, treating problem categories as arms and using absolute advantage as a learning-gain signal, but this requires maintaining a separate bandit policy and predefined categories. Zhang et al. (2025) address the related "difficulty shift" problem by periodically re-estimating difficulty, yet their re-estimation runs on a fixed schedule independent of model performance and they do not study the interaction with on-policy estimators such as RLOO.

3 Method

Our extension adds a curriculum controller around an otherwise standard RLOO trainer. It has three components: an offline difficulty-estimation pass, a bucketing scheme, and an online gating rule.

Difficulty estimation. Before curriculum training begins, we estimate the difficulty of every training prompt x using the frozen post-SFT checkpoint. For each prompt we sample $K{=}16$ rollouts and record the mean verifier reward

$$\bar{r}(x) \;=\; \frac{1}{K}\sum_{i=1}^{K} R\Big(y^{(i)}, x\Big), \qquad y^{(i)} \sim \pi_{\text{SFT}}(\cdot \mid x),$$

as a proxy for how hard the prompt is for the current model. The resulting distribution is heavily right-skewed (Figure 3): roughly a third of prompts sit below $\bar{r} < 0.15$, and on most prompts the SFT model solves fewer than 5% of rollouts, leaving little gradient signal under uniform sampling.

Bucketing into Easy / Medium / Hard. We sort prompts by $\bar{r}(x)$ and split them into three tiers under two schemes:

- **Percentile** (primary): equal-sized tiers at the 33rd and 67th percentiles of $\bar{r}(x)$, which guarantees balanced training pools regardless of the SFT model's overall accuracy.
- **Threshold** (ablation): fixed reward cutoffs at $\bar{r} = 0.2$ and $\bar{r} = 0.5$, which produce imbalanced tiers — a small Easy set and a large Hard set — because the difficulty distribution is skewed.

Table 1 reports the resulting tier sizes and cutoffs. The percentile scheme yields three roughly equal pools ($\sim$6.1k prompts each), whereas the threshold scheme leaves only 2,892 Easy prompts against 8,148 Hard ones.

	Percentile			Threshold		
Tier	Samples	%	$\bar{r}$ cut	Samples	%	$\bar{r}$ cut
Easy	6,242	33.9	≥ 0.375	2,892	15.7	≥ 0.5
Medium	6,054	32.8	≥ 0.144	7,392	40.1	≥ 0.2
Hard	6,136	33.3	—	8,148	44.2	—

Table 1: Tier sizes and $\bar{r}(x)$ cutoffs under both bucketing schemes. The percentile split is balanced by construction; the threshold split is imbalanced because the difficulty distribution is right-skewed.

Adaptive gating. Training starts on the Easy tier only. A controller tracks the rolling pass rate on a held-out validation slice of the currently active tier over the last $M{=}25$ steps. The next tier unlocks when that pass rate clears a tier-specific threshold,

$$\tau_{\text{easy}} = 0.5, \qquad \tau_{\text{medium}} = 0.3,$$

at which point the new tier's prompts are *added* to the sampling pool, so the model continues training on all previously unlocked tiers rather than swapping them out. We hold out 15% of each tier purely for gating; those prompts are never trained on, which keeps the transition decision from being conflated with optimization progress on the training prompts themselves. Algorithm 1 summarizes the loop.

Algorithm 1 Adaptive success-rate-gated curriculum for RLOO

0: Score each prompt x with $\bar{r}(x)$ from frozen π_{SFT}; bucket into Easy/Medium/Hard
0: Hold out 15% of each tier for gating; initialize active pool $\leftarrow$ Easy (train split)
0: **for** step $t = 1 \dots T$ **do**
0: Sample a batch of prompts from the active pool; run RLOO update on π_θ
0: **if** next tier exists **and** rolling pass rate on active tier's val slice (last M steps) $> \tau$ **then**
0: Add next tier's train split to the active pool
0: **end if**
0: **end for**=0

Relative-improvement gating. A weakness of absolute thresholds is that they require knowing the right τ in advance; if set too low the gate never binds. We therefore experiment with a *plateau-based* variant that is self-calibrating. Let p_0 be the pass rate at the first eval of a tier and p_t the pass rate at eval t. A tier unlocks when two conditions hold simultaneously:

1. **Minimum relative gain:** $(p_t - p_0)/\max(p_0, \epsilon) \geq \gamma$, ensuring the model has learned something from the tier before advancing.
2. **Plateau detected:** the per-eval improvement $\Delta_t = p_t - p_{t-1} < \delta$ for P consecutive evals, indicating the model has extracted most of the available learning signal.

We set $\gamma{=}0.05$, $\delta{=}0.015$, $P{=}3$, require at least 4 evals before considering unlock, and evaluate every 3 steps (instead of 5) for faster responsiveness. Because the gate adapts to wherever the model starts, no task-specific threshold tuning is needed.

Novelty. The novel components are two gating rules — one absolute-threshold-based and one plateau-based — that drive tier transitions using the policy's own real-time validation pass rate, computed from the verifier signal that RLOO already produces, with no auxiliary model and no predefined timetable. This isolates the effect of *adaptivity* from the effect of *ordering*, which we test directly against fixed-schedule curricula in our experiments.

4 Experimental Setup

Task and verifier. We use Countdown (**?**): given 3–4 numbers and a target, the model must produce an arithmetic expression evaluating to the target. The rule-based verifier follows TinyZero (**?**), awarding 0.0 for an unparseable answer, 0.1 for a correctly formatted but incorrect answer, and 1.0 for a correct expression that uses the provided numbers validly.

Models and data. We initialize from the SFT checkpoint: `ba144220/cs224r-default-project-sft` (a fine-tune of Qwen2.5-0.5B-Base). All six experiments train on an *identical* fixed 18,432-prompt subset (`ba144220/countdown_tasks_3to4_subset_18k`) so that any difference is attributable to the curriculum and not to which prompts were seen. The subset has zero overlap with the 50-prompt held-out test set. Pre-computed difficulty scores are released as `ba144220/countdown_difficulty_scores`.

Training details. RLOO with batch size 128 for 120 steps. Following the project sampling criterion, training rollouts use temperature 1.0 with top-k, top-p, and min-p disabled; evaluation uses temperature 0.6, top-k 20, top-p 0.95. The SFT initialization itself was trained for 6 epochs at learning rate 5×10^{-5}, batch size 64 (gradient accumulation 8), cosine schedule with 5% warmup, weight decay 0.01, and gradient clipping at 1.0 on a single H100 via Modal.

Baselines and conditions. We compare six conditions on the same prompts and budget:

1. **Vanilla RLOO** — uniform random sampling (primary baseline).
2. **Fixed Curriculum (Pct.)** — add Medium at step 40 and Hard at step 80 *unconditionally*, using the percentile tiers. This isolates ordering from adaptivity.
3. **Sorted Fixed Curriculum** — sort all prompts by difficulty (easiest first) and linearly expand the sampling pool over 120 steps. No discrete tiers; a smooth difficulty ramp that tests whether tier boundaries are necessary.
4. **Adaptive Curriculum (Pct.)** — performance-gated curriculum on the balanced percentile tiers. Tiers unlock when rolling validation pass rate exceeds a threshold (τ_{easy}=0.5, τ_{medium}=0.3).
5. **Adaptive Curriculum (Thr.)** — the same gating applied to the imbalanced threshold tiers (ablation isolating the effect of tier balance).
6. **Adaptive Relative (Pct.)** — plateau-based gating on percentile tiers. Instead of absolute thresholds, tiers unlock when relative improvement over the baseline plateaus (relative gain $\geq 5\%$ and per-eval delta <0.015 for 3 consecutive evals). Eval every 3 steps instead of 5 for faster responsiveness.

Metrics. We report test pass@k for $k \in \{1, 4, 8, 16\}$ and test mean reward at the final checkpoint, the standard RLOO training-side diagnostics (training mean reward and the fraction of zero-reward rollouts), and two curriculum-specific quantities: the rolling gating pass rate and the step at which each tier unlocks. Pass@k measures functional correctness, which is the probability that at least one of k sampled completions passes the verifier, and is the headline downstream metric; mean reward captures partial-credit formatting behavior; the zero-reward fraction directly measures wasted gradient compute.

5 Results

5.1 Quantitative Evaluation

Figure 1 and Table 2 report final-checkpoint test accuracy. The adaptive percentile curriculum (Exp. 4) is the strongest condition at every k, reaching pass@1 $= 0.660$ and pass@16 $= 0.840$ — gains of $+2.9$ and $+2.0$ points over vanilla RLOO — and the highest test mean reward (0.666 vs. 0.659).

Method	pass@1	pass@4	pass@8	pass@16	mean reward
1. Vanilla RLOO	0.631	0.731	0.773	0.820	0.659
2. Fixed Curr. (Pct.)	0.637	0.725	0.757	0.800	0.646
3. Sorted Fixed	0.628	0.717	0.754	0.800	0.633
4. **Adaptive (Pct.)**	**0.660**	**0.772**	**0.807**	**0.840**	**0.666**
5. Adaptive (Thr.)	0.655	0.745	0.781	0.820	0.661
6. Adaptive Rel. (Pct.)	0.646	0.737	0.770	0.820	0.646

Table 2: Final-checkpoint test accuracy (pass@k) and mean reward on the 50-prompt test set. Best per column in bold. The adaptive percentile curriculum (Exp. 4) dominates every metric.

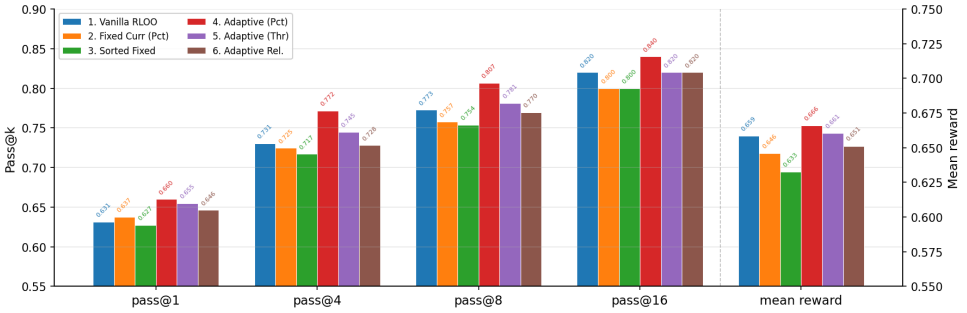
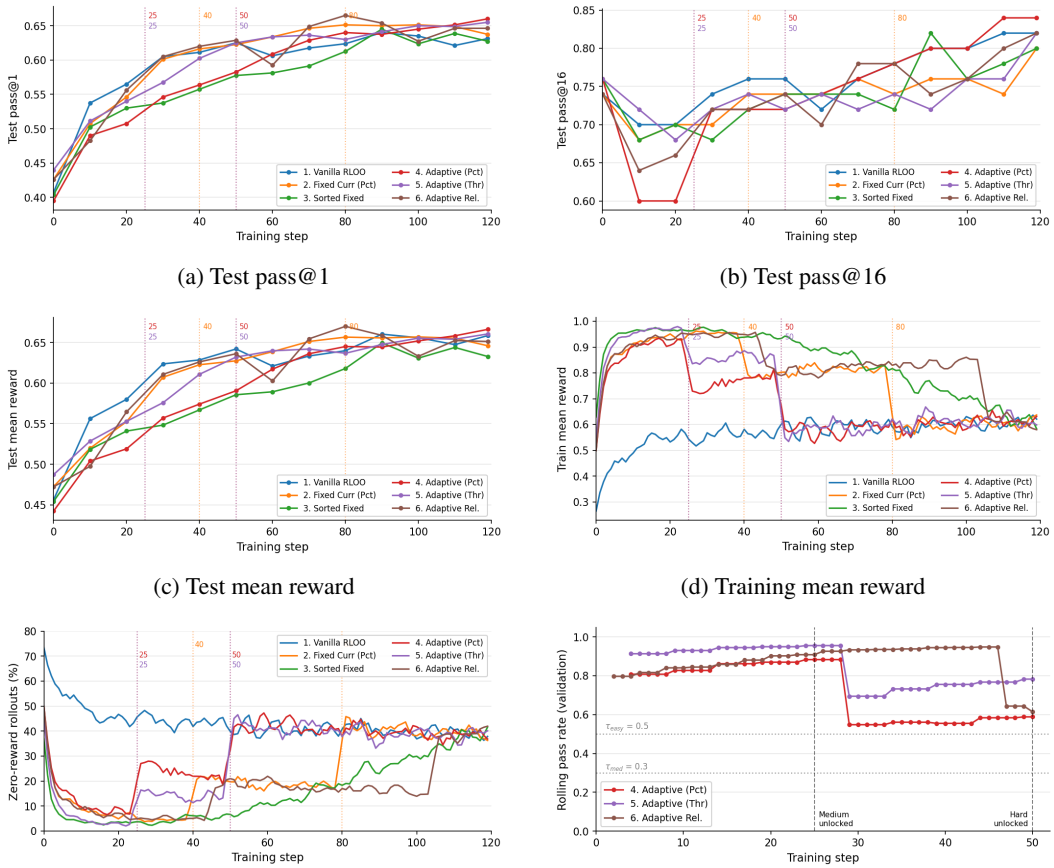


Figure 1: Test accuracy (pass@k) and mean reward at the final checkpoint across all six conditions. The adaptive percentile curriculum (Exp. 4) is highest everywhere.

Key comparisons answer our research questions:

- **Exp. 2 vs. 1 — fixed ordering alone does not help.** The fixed curriculum essentially ties vanilla at pass@1 (0.637 vs. 0.631) and is actually *worse* at higher k (0.800 vs. 0.820 at pass@16). Easy-to-hard ordering by itself is not the source of any gain.
- **Exp. 3 vs. 2 — smooth ramp does not help either.** The sorted fixed curriculum (0.628 pass@1) performs comparably to the discrete fixed curriculum, confirming that neither tier boundaries nor a continuous difficulty ramp improve over vanilla RLOO without performance gating.
- **Exp. 4 vs. 2, 3 — adaptive gating is the active ingredient.** Holding the tiers fixed and changing only the transition rule from a fixed schedule to performance gating moves pass@1 from 0.637 to 0.660 and pass@16 from 0.800 to 0.840. Adaptivity, not ordering per se, drives the improvement.
- **Exp. 4 vs. 5 — balance matters.** The balanced percentile split beats the imbalanced threshold split at every k. A starved Easy tier (only 2,892 prompts under the threshold scheme) undercuts the curriculum's early phase.
- **Exp. 6 vs. 4 — relative gating slightly underperforms.** The plateau-based relative-improvement gating (0.646 pass@1) falls short of the absolute-threshold gating (0.660). The unlock timings explain the gap: Exp. 4 unlocked Medium at step 24 and Hard at step 49, giving 71 steps of training on hard prompts; Exp. 6 unlocked Medium at step 44 and Hard at step 104, leaving only 16 steps on the hardest tier. Ironically, the original thresholds being too low was an advantage here: early unlock gave Exp. 4 far more exposure to difficult prompts.

Sample efficiency and wasted compute. Figure 2 traces the training-side diagnostics. Vanilla RLOO begins with more than 70% of rollouts earning zero reward and stays above 40% throughout (panel e); the adaptive curricula cut this sharply by concentrating early training on solvable prompts. As a direct consequence, curriculum runs collect more reward per step across training (panel d), and the test-side curves (panels a–c) reach higher accuracy within the same 120-step budget.



(a) Test pass@1

(b) Test pass@16

(c) Test mean reward

(d) Training mean reward

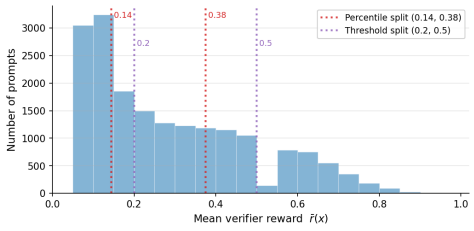
(e) Zero-reward rollout fraction

(f) Gating pass rate

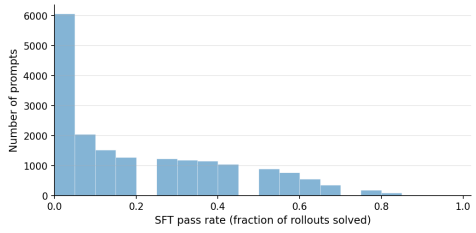
Figure 2: Training and test curves over 120 RLOO steps. Curriculum runs reduce the zero-reward rollout fraction (e) and accumulate more reward per step (d), translating into higher test accuracy (a–c) at equal budget. Panel (f) shows the rolling gating pass rate that drives tier transitions.

5.2 Qualitative Analysis

Difficulty structure of the data. Figure 3 explains *why* curriculum helps here. The per-prompt mean-reward distribution (a) is sharply right-skewed, and the per-tier SFT pass rate (b) shows that under the SFT model most prompts produce successful rollouts only a few percent of the time. Uniform sampling therefore spends most of its budget on prompts that return no gradient; sorting by $\bar{r}(x)$ and starting easy targets the small slice of prompts that actually sit on the competence frontier.



(a) Prompt difficulty distribution



(b) Per-tier SFT pass rate

Figure 3: Difficulty structure of the training set under the SFT model. The reward distribution is heavily right-skewed (a), and most prompts are solved on only a small fraction of rollouts (b), leaving little gradient signal under uniform sampling.

Gating behavior and a key failure mode. The gating-pass-rate curve (Figure 2f) reveals that both thresholds were set too low: the active tier's rolling pass rate cleared τ almost immediately, so each tier unlocked at the earliest moment the minimum window allowed. In practice the adaptive schedule therefore collapsed to a fixed 25-step-per-tier cadence. That the adaptive runs still beat the explicitly fixed curriculum (Exp. 2) is notable — it suggests the *shape* of even a near-minimum gated schedule (unlock-as-soon-as-ready, anchored to the validation slice) is better than hand-picked unlock steps — but it also means we have likely *under*-measured the benefit of true adaptivity. Properly tuned thresholds that make the gate bind would let the curriculum dwell longer on tiers the policy has not yet mastered.

Sample rollout. The SFT initialization already produces well-structured chains of thought; the example below (scored 1.0 by the verifier) illustrates the format the curriculum-trained policies continue to refine.

Sample Countdown rollout

Prompt: Using the numbers [44, 19, 35], create an equation that equals 98.

Model output:
<think>
Let me try to find a path to 98.
First, the numbers are 44, 19, and 35.
$44 + 19 = 63$ (getting closer)
$63 + 35 = 98$ (this is the target)
Verify: $(19 + 44) + 35 = 98$ ✓
</think>
<answer> $(19 + 44) + 35$ </answer>

Score: 1.0 (correct)

6 Discussion

The headline gains are modest in absolute terms (single-digit points), but they come for free: the curriculum reuses the verifier signal already in the pipeline, adds no model and no data, and *reduces* compute by cutting zero-reward rollouts. The clean Exp. 2/3 vs. 4 comparison is the most informative result — it shows that the benefit is specifically about *when* harder prompts are introduced relative to the policy's competence, not about easy-to-hard ordering in the abstract. The chief difficulty we encountered was tuning the gating thresholds: because we set them conservatively, the gate rarely bound and we likely observed only a lower bound on what adaptive pacing can deliver. A broader impact note: difficulty-aware curricula could in principle amplify whatever biases exist in the verifier's notion of "easy," which matters more for open-ended tasks than for a closed-form arithmetic verifier like Countdown's.

7 Conclusion

On Countdown, an adaptive, success-rate-gated curriculum improves RLOO across every pass@k metric while sharply reducing wasted gradient computation, and a controlled ablation attributes the gain to performance gating rather than to ordering alone. The take-home message is that *how a curriculum decides to advance* matters more than the mere fact of going easy-to-hard.

Here are some future directions:

- **Online difficulty re-estimation.** $\bar{r}(x)$ is measured once from the SFT model and goes stale as the policy improves. Periodically re-scoring prompts would keep tier assignments aligned with the model's actual competence frontier.
- **Continuous frontier-aware sampling.** Three discrete tiers are a coarse approximation. A sampler that weights each prompt by its current learnability — e.g., proportional to reward variance across rollouts — could allocate gradient more efficiently than hard tier boundaries.
- **Learned or tighter gating thresholds.** Our absolute thresholds were too low and the plateau-based variant unlocked Hard too late (step 104 vs. 49). Learning the gating parameters

online, or combining both signals (minimum absolute bar plus plateau detection), could recover the best of both approaches.

- **Generality.** All results are on a single task (Countdown) and a single base model (Qwen2.5-0.5B). Validating on additional reasoning tasks and larger models is the natural next step.

8 Team Contributions

- **Yuchi Hsu:** Implemented SFT and the RLOO trainer; ran difficulty scoring, all six training conditions, and final evaluation; produced the training and results figures; assisted in writing
- **Ryan He:** Implemented IPO and difficulty estimator; Designed the curriculum extension and gating rule; assisted in writing

References

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

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

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

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

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research* 21, 181 (2020), 1–50.

Shubham Parashar et al. 2025. Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning. *arXiv preprint arXiv:2506.06632* (2025).

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020. Automatic Curriculum Learning for Deep RL: A Short Survey. *arXiv preprint arXiv:2003.04664* (2020).

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. *arXiv preprint arXiv:2505.08364* (2025).