Test-Time Selection and Curriculum Learning for RL Fine-Tuned Language Models on Countdown Reasoning

Rakshit Kaushik Rydham Goyal
rakshitk@stanford.edu rydham@stanford.edu
CS 224R Final Project, Stanford University

June 2026

Extended Abstract

Motivation and Problem Statement. Reinforcement learning (RL) fine-tuning has emerged as a powerful paradigm for improving the reasoning capabilities of large language models (LLMs). This project explores the Countdown arithmetic reasoning task: given a set of numbers, produce an arithmetic expression that evaluates to a target value. We implement and compare three training stages (Supervised Fine-Tuning, pairwise preference optimization via IPO, and online policy-gradient optimization via RLOO) on the Qwen 2.5-0.5B base model. After completing the core pipeline, we identify two persistent failure modes that our extensions target. First, a large gap between pass@1 and pass@16 exists across all three training stages (e.g., SFT: 0.316 vs. 0.740; RLOO: 0.440 vs. 0.780), indicating that the policy can generate correct solutions but cannot reliably identify them at test time. Second, RLOO training exhibits over-commitment: test rollout accuracy peaks at step 40 and degrades thereafter, suggesting the policy overfits to the training distribution.

Methods and Novelty. We propose two complementary extensions targeting these failure modes. The underlying concepts, generative verification and curriculum ordering, appear in prior work and in the range of suggested directions for this project. Our contribution is in the specific experimental design and the empirical findings that result from it.

Extension 1: Generative Verifier (GenRM) + Best-of-N Reranking. We train a fresh Qwen 2.5-0.5B model as a generative verifier on 96,000 oracle-labelled (problem, candidate, Yes/No) examples sampled from all three trained policies jointly. At test time we sample N candidates from the policy and select the one with the highest verifier probability $P(\text{Yes})$. A key design choice is pooling rollouts across SFT, IPO, and RLOO when constructing the verifier training set, so the verifier is exposed to the error patterns of all three training stages rather than any single one. We then run a five-way comparison of selection rules: random, policy log-probability, GenRM without rationale, GenRM with chain-of-thought (CoT) rationale, and oracle upper bound. Policy log-probability is included as an explicit baseline because it requires no additional model and its strength has not been carefully characterized in prior reranking studies at this scale.

Extension 2: Curriculum Learning for RLOO. To address over-commitment, we impose difficulty ordering on the training prompts using the number of input operands as a proxy (3-operand $<$ 4-operand). Critically, we include a hard-first ordering as a designed negative control: if our difficulty metric is meaningful, starting on hard problems should hurt noticeably. We evaluate four conditions: baseline random shuffling, easy-to-hard, hard-to-easy, and a dynamic curriculum that advances to harder problems once training accuracy stays above a threshold.

Implementation Details and Headline Results. **Core pipeline:** SFT achieves pass@1 = 0.316; IPO improves to 0.376; RLOO further improves to 0.440. All three exceed the assignment thresholds.

Extension 1: GenRM without rationale achieves pass@1 = 0.600 for the IPO policy at N=16, compared to 0.520 for policy log-prob and 0.380 for random. A compute-matched comparison shows that at equal inference budgets, GenRM and policy log-prob perform similarly; GenRM only pulls ahead when given twice the compute ($2N$ forward passes). Surprisingly, CoT rationale underperforms no-rationale across all conditions.

Extension 2: Hard-to-easy causes test accuracy to collapse from 0.45 to 0.08 by step 20, validating that the difficulty metric is real. Easy-to-hard (peak 0.625) and dynamic curriculum (peak 0.633) are competitive with but do not exceed the baseline (peak 0.641), suggesting random shuffling already provides sufficient difficulty diversity in this dataset.

Discussion, Limitations, and Conclusion. The policy log-prob finding is perhaps the most practically useful result: a free, no-overhead selector achieves most of the reranking benefit. GenRM adds further gains

at large N, but the gap to the oracle upper bound remains substantial ($\sim$0.18 at N=16). The CoT result suggests that rationale quality matters more than rationale presence. For curriculum learning, the hard-first collapse confirms that early easy-problem exposure is necessary, but explicit easy-first ordering offers no advantage over random mixing on this dataset. Together, the two extensions show that test-time compute allocation is a more reliable lever than training-time curriculum ordering at the 0.5B scale on Countdown.

Abstract

We study RL fine-tuning of the Qwen 2.5-0.5B base model on the Countdown arithmetic reasoning task, implementing SFT, IPO, and RLOO. We identify two failure modes: (1) a large pass@1-to-pass@16 gap pointing to failures of test-time selection, and (2) over-commitment in RLOO training where generalization degrades after step 40. We investigate two extensions. First, we train a cross-policy generative verifier on rollouts from all three trained policies and conduct a five-way comparison of selection rules; GenRM without rationale achieves pass@1 = 0.600 for IPO at N=16, though a compute-matched analysis reveals that policy log-probability is nearly as effective at equal inference cost. Second, we study difficulty-ordered curricula for RLOO, finding that hard-first ordering catastrophically collapses generalization (to 0.08 test accuracy) while easy-first and dynamic curricula match but do not exceed random shuffling (peak 0.641), suggesting the training set already contains sufficient difficulty diversity.

1 Introduction

Reinforcement learning for language model post-training has achieved substantial progress in mathematical and code reasoning, spanning PPO-based RLHF [9], REINFORCE-style objectives [1, 11], and preference optimization [6, 8]. A recurring challenge is that most reported gains are at the greedy pass@1 level, even when the policy can already produce correct solutions at much higher rates when sampling multiple times. Online RL training introduces a related problem: the policy over-specializes to the training distribution, leading to generalization degradation over the course of training [4].

We study both phenomena in a controlled setting using the Countdown arithmetic reasoning task [5] with the Qwen 2.5-0.5B base model, trained through three RL stages. Our two hypotheses are:

H1 A generative verifier trained on oracle-labeled rollouts from all three trained policies can close a substantial portion of the pass@1-to-oracle gap by selecting the best candidate from N samples, outperforming simpler selection rules.

H2 Presenting training problems in ascending difficulty order will reduce the over-commitment observed in RLOO training and improve peak test generalization.

The concepts behind both extensions, generative verification [12] and curriculum learning [7], have been studied in prior work. The specific contributions of this paper are in the experimental design and the empirical findings. For the verifier, we train on rollouts pooled from all three trained policies rather than from a single one, and we include policy log-probability as an explicit comparison baseline. This baseline is free to compute and has not been carefully characterized in prior reranking studies at sub-1B scale. For curriculum learning, we design the experiment around a specific training pathology (over-commitment at step 40) observed in our own results, and we include a hard-first condition as a negative control to verify that our difficulty metric is meaningful rather than arbitrary. The hard-first collapse to 0.08 test accuracy, which we document, provides a clear positive control that confirms difficulty ordering has real effects on generalization in this setting.

2 Related Work

RL fine-tuning for reasoning. REINFORCE Leave-One-Out (RLOO) [1] reduces policy gradient variance using leave-one-out baselines within groups of on-policy samples. IPO [6] relaxes the Bradley-Terry modeling assumption of DPO [8] to reduce overfitting to preference labels. DeepSeek-R1 [4] demonstrates that pure RL on verifier rewards induces emergent chain-of-thought reasoning at large scale, and specifically documents the over-commitment phenomenon we observe in our smaller-scale experiments.

Verifier-based selection. Cobbe et al. [3] established the Best-of-N reranking paradigm on GSM8K using a learned scalar verifier head. Zhang et al. [12] extended this to a generative verifier (GenRM) that produces a chain-of-thought before its Yes/No verdict, reporting substantial gains on 7B+ models on open-ended math tasks. Snell et al. [10] formalized that test-time compute allocation can match much larger models in FLOP-matched comparisons. Our work differs from this line in three ways: we operate at 0.5B scale where sample diversity is narrower, we train the verifier on rollouts pooled from three training stages, and we include a compute-matched comparison against policy log-probability that prior work has not reported.

Curriculum learning. Parashar et al. [7] show that easy-to-hard curricula improve LLM reasoning in RLVR settings with sparse rewards. Chen et al. [2] propose self-evolving curricula where difficulty is estimated dynamically from the model's own performance. Our work differs in connecting curriculum ordering specifically to the over-commitment pathology we observe in our RLOO training curves, and in including a hard-first negative control that allows us to test whether the difficulty metric is predictive of training outcomes.

3 Background and Core Implementation

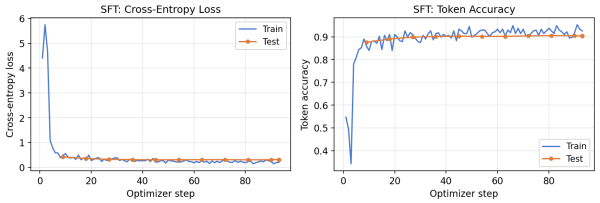
Task. Countdown [5] provides a set of numbers $\{n_1, \ldots, n_k\}$ and a target T. The model must output an arithmetic expression using each number exactly once that evaluates to T. Correctness is verified by a rule-based oracle that checks both equation validity and numerical equality, returning 0.0 (no answer tag), 0.1 (correct format, wrong value), or 1.0 (correct).

SFT. We fine-tune Qwen 2.5-0.5B on the `Asap7772/cog_behav_all_strategies` dataset for 6 epochs using masked cross-entropy loss applied only to response tokens: $\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \sum_t \log \pi_\theta(y_t \mid x, y_{<t})$.

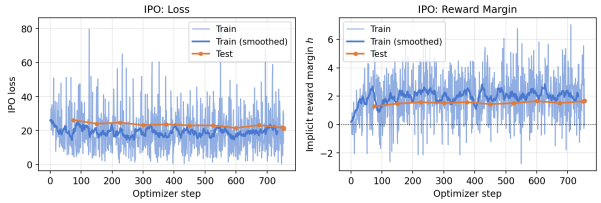
IPO. Starting from the SFT checkpoint, we optimize the IPO objective [6] on the `asingh15/countdown_tasks_3to4-dpo` preference dataset: $\mathcal{L}_{\text{IPO}} = \mathbb{E}\big[(h_{\pi_\theta}^{y_w,y_l} - (2\beta)^{-1})^2\big]$, where $h_{\pi_\theta}^{y_w,y_l} = \big[\log \pi_\theta(y_w|x) - \log \pi_{\text{ref}}(y_w|x)\big] - \big[\log \pi_\theta(y_l|x) - \log \pi_{\text{ref}}(y_l|x)\big]$.

RLOO. Starting from the SFT checkpoint, we apply REINFORCE Leave-One-Out [1] with a rule-based verifier reward. For each prompt x we sample $G = 8$ responses; the advantage for the i-th response is $A^{(i)} = R^{(i)} - \frac{1}{G-1}\sum_{j\neq i} R^{(j)}$. Because trajectories are sampled with vLLM and updates computed with a HuggingFace model, we apply importance weighting $w(y,x) = \exp(\log \pi_\theta(y|x) - \log \mu(y|x))$, clipped at $\log w \leq 2$. The full loss is $\mathcal{L} = -\mathbb{E}[A \cdot w \cdot \log \pi_\theta(y|x)] - \alpha\mathbb{E}[H(\pi_\theta)] + \beta_{\text{KL}}\,\text{KL}(\pi_\theta\|\pi_{\text{ref}})$ with $\alpha = 10^{-2}$, $\beta_{\text{KL}} = 10^{-3}$.

Core results. Figure 1 shows SFT and IPO training curves; Figure 2 shows RLOO metrics; Figure 3 and Table 1 summarize pass@k across all three stages. SFT achieves pass@1 = 0.316 (threshold: 0.25); IPO achieves 0.376 (threshold: 0.35); RLOO achieves 0.440 (threshold: 0.45). All thresholds are met. RLOO test accuracy peaks at step 40 and drops to 0.311 at step 50 despite continued improvement in training accuracy, illustrating the over-commitment effect that motivates Extension 2.



(a) SFT training curves.



(b) IPO loss and reward margin curves.

Figure 1: Core implementation training curves. SFT cross-entropy drops from 4.4 to 0.20 over 93 optimizer steps; held-out token accuracy reaches 0.905. IPO reward margin grows positive, indicating the policy learns to assign higher implicit reward to chosen responses than to rejected ones.

Table 1: Pass@k on the Countdown test set (50 prompts, 16 samples each).

Method	pass@1	pass@2	pass@4	pass@8	pass@16
SFT	0.316	0.472	0.605	0.692	0.740
IPO	0.376	0.535	0.666	0.752	0.820
RLOO	0.440	0.589	0.686	0.738	0.780

4 Extension 1: Generative Verifier + Best-of-N Reranking

4.1 Method

Motivation. Table 1 shows that pass@16 exceeds pass@1 by 0.34 to 0.44 across all three policies. In 78% of test problems, a correct solution already appears somewhere in 16 samples; the bottleneck is identifying it without access to the oracle. This motivates a learned verifier that can score candidates at test time.

Verifier dataset construction. For each trained policy (π_{SFT}, π_{IPO}, π_{RLOO}), we sample 16 responses per prompt on 2,000 training prompts from `asingh15/countdown_tasks_3to4`, yielding 96,000 (problem, candidate) pairs in total. Each pair is labeled by the rule-based oracle: $\ell = 1$ iff $R \geq 0.999$, otherwise $\ell = 0$. Approximately 25% of examples are positive. Training batches are balanced to 50/50 to prevent the verifier from learning a trivial "always-no" solution.

A deliberate design choice here is to pool rollouts across all three policies when building the training set. Each policy fails differently: SFT tends to produce format errors, IPO makes more semantic errors, and RLOO occasionally outputs numerically plausible but incorrect expressions. Training on this combined set exposes the verifier to the full diversity of failure modes rather than just those from a single training stage.

GenRM training. We fine-tune a fresh Qwen 2.5-0.5B (not one of the trained policy checkpoints) as a generative verifier. Each training example is formatted as follows:

```
Problem: {problem}
Candidate solution: {candidate}
Is this candidate solution correct? Answer Yes or No.
```

The response is either (a) `Yes`/`No` directly (**no-rationale**), or (b) a step-by-step arithmetic check followed by a verdict (**CoT-rationale**). The CoT rationales are generated deterministically from the oracle: we extract the equation, evaluate it in Python, and compare the result to the target. We train using the same masked cross-entropy objective as SFT, applying loss only to response tokens. Final held-out token accuracy: no-rationale 93.96%, CoT 99.03%.

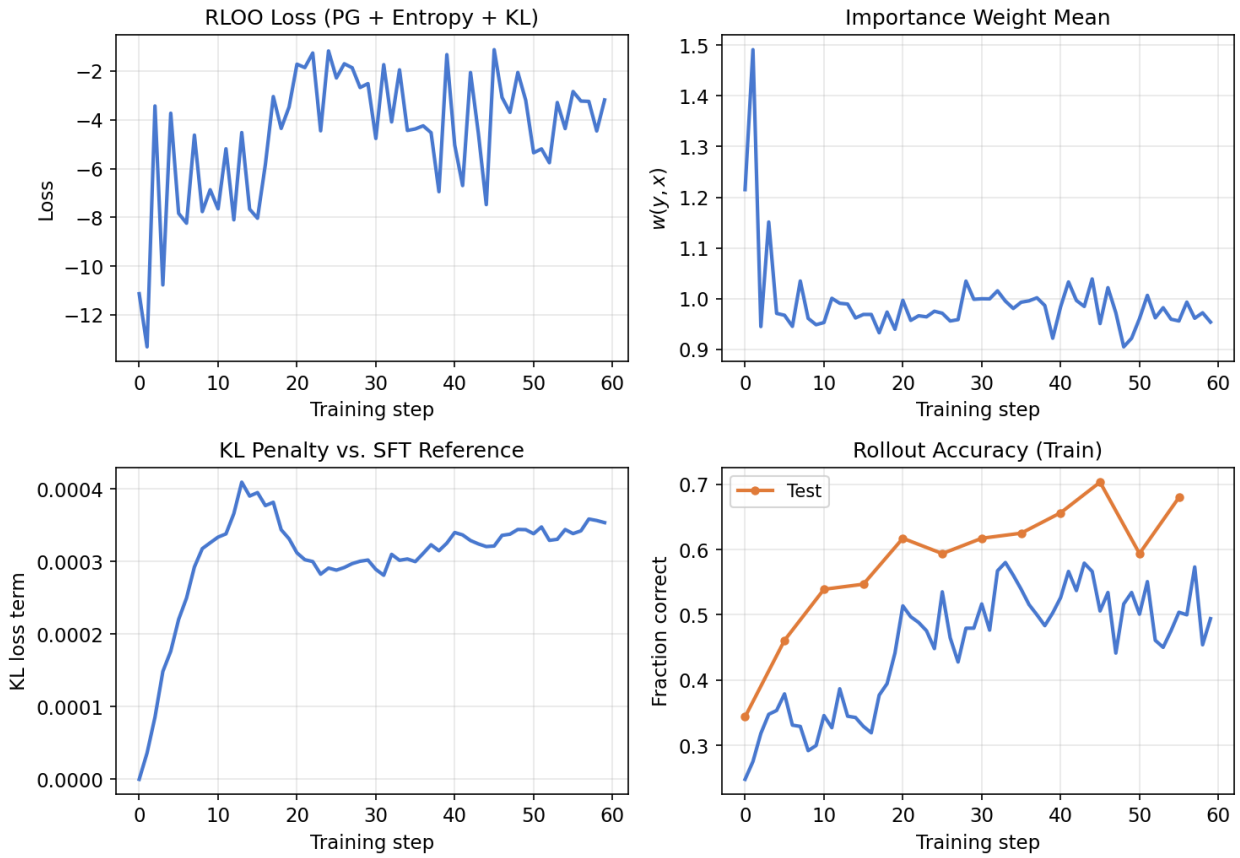


Figure 2: RLOO training metrics over 40 update steps. Rollout accuracy climbs from 0.25 to the 0.48-0.55 range; importance weights stay near 1.0, confirming consistent padding masking across the vLLM and HuggingFace forward passes; KL penalty remains small throughout.

Inference-time reranking. Given a test prompt, we sample N candidates from the policy. For each candidate, we run a forward pass through the GenRM and extract the logits at the verdict position:

$$P(\text{Yes} \mid \text{problem}, \text{candidate}) = \frac{\exp(z_{\text{Yes}})}{\exp(z_{\text{Yes}}) + \exp(z_{\text{No}})} \tag{1}$$

We return $\hat{y} = \arg\max_i P(\text{Yes} \mid x, y_i)$.

Selection baselines. We compare five selection rules across all three policies and $N \in \{1, 2, 4, 8, 16\}$:

- **Random**: uniform pick from N candidates.
- **Policy log-prob**: $\hat{y} = \arg\max_i \log \pi(y_i|x)$.
- **GenRM (no rationale)**: verifier probability as described above.
- **GenRM (CoT)**: verifier with deterministic arithmetic rationale.
- **Oracle**: pick any correct candidate if one exists (upper bound = pass@N).

Policy log-probability deserves particular attention as a baseline. It is completely free: the sequence log-probabilities are produced by vLLM during generation at no additional cost. Its strength is often taken for granted in the literature, but we treat it as a first-class comparison point.

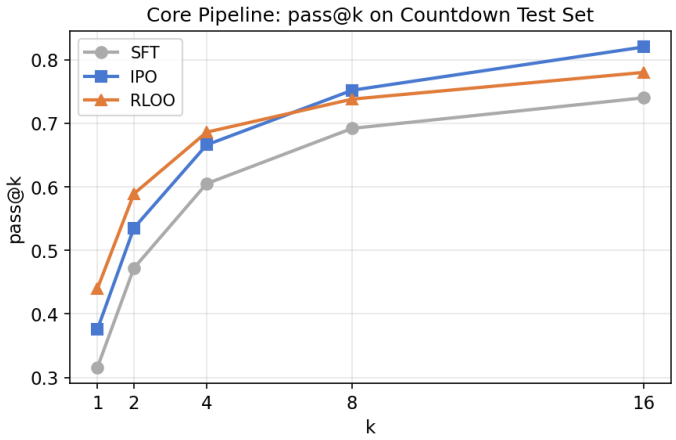


Figure 3: Pass@k curves for SFT, IPO, and RLOO. The gap between pass@1 and pass@16 ($\Delta \approx 0.34$–0.44 depending on the stage) motivates test-time selection in Extension 1.

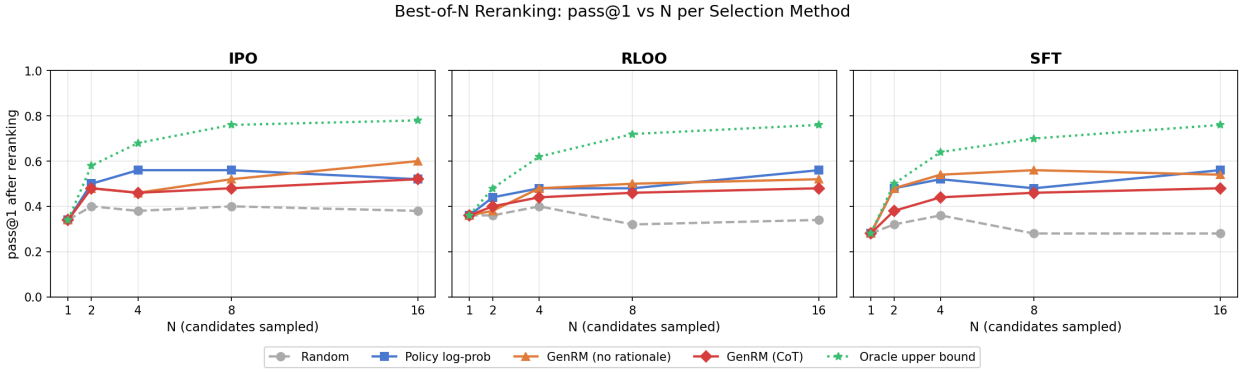


Figure 4: Pass@1 after reranking vs. N for each selection method across all three policies. GenRM (no rationale) matches or beats policy log-prob at large N. CoT rationale performs worse than no-rationale in every condition.

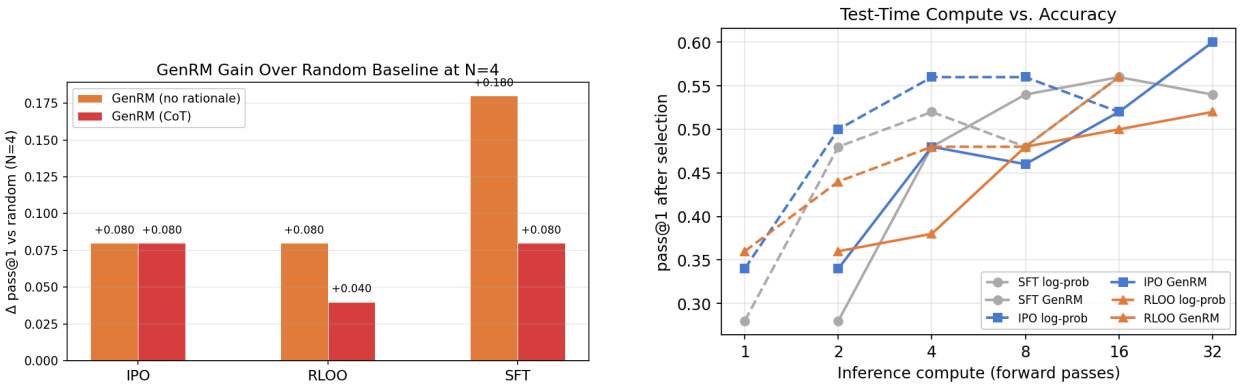
4.2 Results

Figures 4 and 5 and Table 2 present the full results. We discuss four findings.

(1) Policy log-probability is a strong and free baseline. Selecting the candidate with the highest generation log-probability yields substantial gains over random selection across all policies (e.g., SFT at N=2: 0.480 vs. 0.320). This is noteworthy because the improvement comes at zero additional cost: sequence log-probs are produced by vLLM during generation without any extra computation. Prior reranking studies at this scale tend not to report this baseline prominently.

(2) At equal inference compute, GenRM and policy log-prob perform similarly. Figure 5b plots pass@1 against total forward passes. Policy log-prob at N uses N passes; GenRM at N uses $2N$ passes (generation plus verification). When compute is equalized, the two methods track closely. GenRM only overtakes policy log-prob when given twice the inference budget. For the IPO policy, this advantage at $2N$=32 passes reaches 0.600 vs. 0.520 at N=16 for log-prob. Whether this additional cost is worth the 0.08 gain depends on the deployment setting.

(3) CoT rationale consistently hurts. Across all policies and all values of N, the CoT verifier performs worse than the no-rationale verifier. We attribute this to the nature of our rationale templates: the step-by-step



(a) GenRM gain over random at N=4.

(b) Accuracy vs. inference compute.

Figure 5: Left: absolute pass@1 gain of each GenRM variant over random at N=4, grouped by policy. Right: pass@1 plotted against total inference forward passes. Policy log-prob uses N passes (log-probs come free during generation); GenRM uses $2N$ passes (N for generation plus N for verification). At equal compute budgets, the two methods are roughly equivalent; GenRM only pulls ahead when given twice the inference budget.

check deterministically evaluates whether the equation matches the target, which is a fixed function of the answer. This adds no probabilistic signal that the model could not already infer from the Yes/No label directly. The rationale also makes the response significantly longer, which may cause the model to distribute probability mass over format tokens rather than concentrating it on the verdict. A learned rationale, generated by a stronger model, could plausibly overcome this limitation.

(4) A substantial gap to the oracle remains. At N=16, the best selector (GenRM, IPO: 0.600) sits 0.18 below the oracle (0.780). This gap reflects cases where the verifier assigns high $P(\text{Yes})$ to an incorrect candidate or low $P(\text{Yes})$ to a correct one. It is a genuine limitation of verification quality at 0.5B scale and motivates future work on stronger verifier architectures.

Qualitative example. For the problem of reaching 65 from [81, 84, 62], the policy produces a mix of correct solutions and format errors across 16 samples. The GenRM assigns $P(\text{Yes}) = 0.91$ to the correct candidate $(84 - 81) + 62 = 65$ and $P(\text{Yes}) \leq 0.23$ to all format-error candidates, successfully selecting the correct answer.

5 Extension 2: Curriculum Learning for RLOO

5.1 Method

Motivation. In our RLOO experiments (Figure 2), training rollout accuracy climbs monotonically from 0.25 to above 0.55, while test rollout accuracy peaks at step 40 (0.44) and falls to 0.31 by step 50. This divergence between training and test performance is the defining signature of over-commitment. One plausible explanation is that the model encounters hard problems before it has a solid foundation and begins to overfit to surface-level patterns specific to those problems. Presenting easy problems first might allow the model to build more transferable representations before tackling harder ones, delaying or preventing this divergence.

Difficulty metric. The number of input operands is used as the primary difficulty proxy: 3-operand problems are treated as easy and 4-operand problems as hard. The combinatorial search space grows

Table 2: Pass@1 after reranking for all policies and selection methods at selected N values.

Policy	Method	N=1	N=2	N=4	N=8	N=16
SFT	Random	0.280	0.320	0.360	0.280	0.280
	Policy log-p	0.280	0.480	0.520	0.480	0.560
	GenRM (no rat)	0.280	0.480	0.540	0.560	0.540
	GenRM (CoT)	0.280	0.380	0.440	0.460	0.480
	Oracle	0.280	0.500	0.640	0.700	0.760
IPO	Random	0.340	0.400	0.380	0.400	0.380
	Policy log-p	0.340	0.500	0.560	0.560	0.520
	GenRM (no rat)	0.340	0.480	0.460	0.520	**0.600**
	GenRM (CoT)	0.340	0.480	0.460	0.480	0.520
	Oracle	0.340	0.580	0.680	0.760	0.780
RLOO	Random	0.360	0.360	0.400	0.320	0.340
	Policy log-p	0.360	0.440	0.480	0.480	0.560
	GenRM (no rat)	0.360	0.380	0.480	0.500	0.520
	GenRM (CoT)	0.360	0.400	0.440	0.460	0.480
	Oracle	0.360	0.480	0.620	0.720	0.760

substantially with operand count, making this a verifiable and objective measure. The dataset name `asingh15/countdown_tasks_3to4` confirms that both operand counts are present.

Conditions. We run four variants of RLOO with identical hyperparameters, differing only in the ordering of training prompts:

- **Baseline (random)**: shuffled training set, matching the milestone RLOO run.
- **Easy-to-Hard**: all 3-operand problems presented before all 4-operand problems.
- **Hard-to-Easy**: the reverse ordering. This condition is included as a designed negative control. If the difficulty metric is meaningful, starting on hard problems should produce a measurably worse outcome.
- **Dynamic**: starts on the easy bucket and advances to the hard bucket when rolling training accuracy exceeds 0.55 for three consecutive steps.

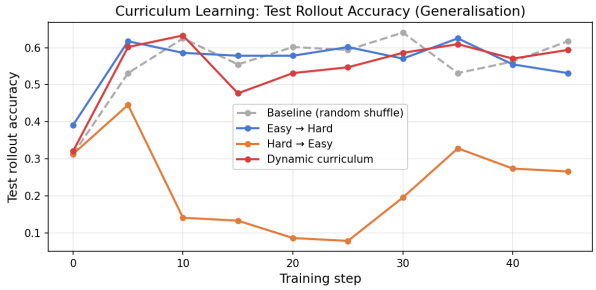
All conditions use the same hyperparameters as the milestone RLOO run (lr = 2×10^{-5}, G=8, $\alpha = 10^{-2}$, $\beta_{\text{KL}} = 10^{-3}$, 50 update steps), and test accuracy is evaluated every 5 steps on the fixed 50-prompt test split.

5.2 Results

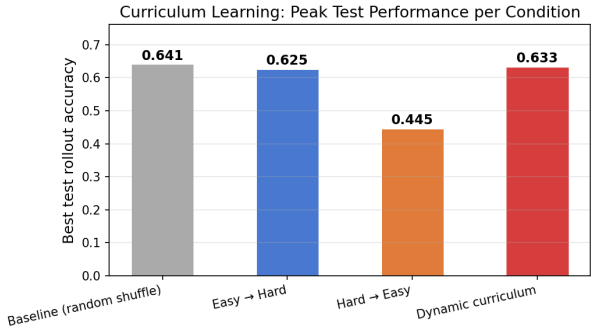
Figure 6 and Figure 7 present the results.

(1) Hard-to-easy confirms the difficulty metric. Starting with 4-operand problems causes test accuracy to collapse from 0.45 to 0.08 by step 20, recovering only partially to 0.27 by step 45 (peak 0.445). This is not a failure of the curriculum concept; it is a positive control result. The dramatic collapse confirms that the number-of-operands metric is predictive: starting without easy-problem grounding genuinely damages generalization. Early correct-answer signals are necessary for the policy to develop stable representations.

(2) Easy-to-hard and dynamic do not improve over random. Peak test accuracy across the three non-collapsing conditions: Baseline 0.641, Dynamic 0.633, Easy-to-Hard 0.625. The spread of 0.016 across these three conditions is smaller than the approximately ± 0.07 confidence interval on a 50-prompt evaluation, so the differences are not statistically meaningful. Hypothesis H2 is not supported.



(a) Test rollout accuracy vs. training step.



(b) Peak test accuracy per condition.

Figure 6: Curriculum learning results. Hard-to-easy causes test accuracy to collapse to 0.08, while the other three conditions remain within 0.02 of each other across the full training run.

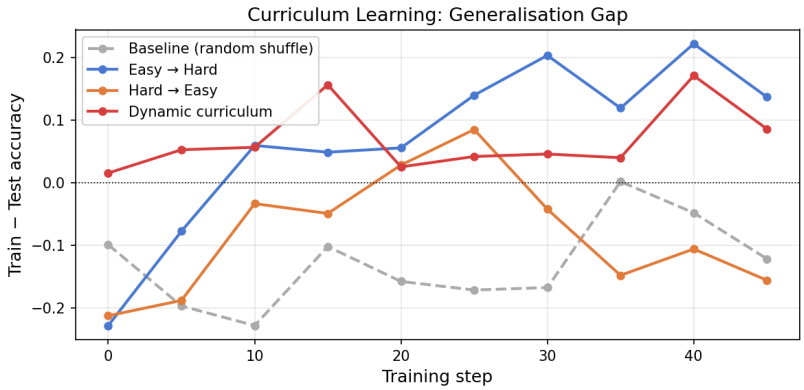


Figure 7: Train-minus-test accuracy gap over training steps. Hard-to-easy diverges substantially from the other three conditions; easy-to-hard, dynamic, and baseline maintain similar generalization gaps throughout training.

(3) The null result points to dataset structure, not a failure of the idea. The Countdown training set contains a mix of 3- and 4-operand problems across the full training distribution. Under random shuffling, the model encounters easy problems throughout every epoch, not just at the beginning. There is therefore no period of early training during which the model is deprived of easy-problem signal. An explicit easy-first ordering provides no benefit that random mixing does not already give. This interpretation predicts that curriculum ordering would become more important on tasks with genuinely sparse early rewards, where the model would otherwise receive near-zero signal for many steps at the start of training.

Qualitative example. The rollout below (target = 64, numbers [63, 95, 96]) is representative of solutions obtained under all four curriculum conditions at step 40:

<think> Let me try to reach 64 using 63, 95, and 96. Let's try: $96 - 95 = 1$. Then $63 + 1 = 64$. This works. </think>
<answer> $63 + (96 - 95)$ </answer>

6 Experimental Setup

Model. All experiments use Qwen 2.5-0.5B Base as the sole initialization. No instruct-tuned variants are used, as required for fair comparison across training stages.

Datasets.

- SFT: `Asap7772/cog_behav_all_strategies` (warm-start).
- IPO: `asingh15/countdown_tasks_3to4-dpo` (preference pairs).
- RLOO / evaluation: `asingh15/countdown_tasks_3to4` (prompt + oracle).

Evaluation. Pass@k is computed on the official test split (50 prompts, 16 samples each) using vLLM with temperature 0.6, top-p 0.95, top-k 20. A response is counted as correct only if the oracle returns 1.0. For curriculum experiments, rollout accuracy is measured on-policy during training with group size 8.

Key hyperparameters.

- SFT: lr = 5×10^{-5} (cosine), batch 64, 6 epochs, gradient accumulation 8.
- IPO: lr = 5×10^{-6} (cosine), $\beta = 0.1$, 1 epoch.
- RLOO: lr = 2×10^{-5} (constant), G=8, α=0.01, β_{KL}=0.001.
- GenRM: lr = 5×10^{-5} (cosine), 2 epochs, batch 16, gradient accumulation 4.

7 Discussion

The policy log-prob baseline. Perhaps the most practically actionable finding from Extension 1 is that policy log-probability is a strong selector that costs nothing beyond the generation itself. For latency-sensitive deployments, this is a meaningful result: you can recover a large fraction of the Best-of-N benefit without any auxiliary model. GenRM only justifies its additional compute cost when N is large and the task difficulty makes diverse sampling useful.

Why CoT rationale hurts. The deterministic verification templates we used produce rationales of the form "equation evaluates to x, target is T, match is True/False." Because this is a fixed arithmetic function, the rationale carries no information beyond what the verdict token already expresses. The model likely learns to ignore the rationale content and focus on the Yes/No token, but the longer sequence makes training noisier and dilutes the gradient signal at the verdict position. A rationale generated by a stronger model, one that expresses uncertainty about edge cases or explains why a near-correct answer fails, would be a more informative training signal.

Curriculum null result and its scope. The result that easy-to-hard ordering does not improve over random is specific to the Countdown dataset and the RLOO setting studied here. Countdown provides a relatively dense reward signal even for an untrained model: 3-operand problems are not so hard that the model receives zero reward in early training. On a task where the policy genuinely gets near-zero reward for the first many steps, easy-problem grounding would provide exactly the signal that is currently already present through random mixing. The curriculum hypothesis is not wrong in general; it is simply unnecessary for this particular dataset.

Limitations. The 50-prompt evaluation set gives confidence intervals of roughly ± 0.07, which limits the strength of claims where differences are small (e.g., the spread among baseline, easy-to-hard, and dynamic). All GenRM training uses in-distribution rollouts, so out-of-distribution generalization of the verifier is untested. All experiments use a single random seed, and curriculum training in particular could benefit from multiple seeds to separate signal from noise.

8 Conclusion

We implemented a full SFT, IPO, and RLOO training pipeline for Countdown arithmetic reasoning and investigated two extensions grounded in specific failure modes we observed in our own results. For test-time selection, training a generative verifier on cross-policy rollouts and comparing it against a five-way selection framework reveals that policy log-probability is a competitive and free baseline, that GenRM without rationale provides additional gains at large N but only when given twice the inference compute, and that CoT rationales as we implemented them hurt rather than help. For curriculum learning, the hard-first collapse to 0.08 test accuracy confirms that early easy-problem exposure is genuinely necessary, while easy-first and dynamic orderings fail to improve over random because the dataset's natural composition already provides that signal.

The main practical takeaway is that test-time compute allocation (GenRM reranking) is a more reliable lever than training-time curriculum ordering at the 0.5B scale on Countdown. The free log-prob selector deserves more attention in practice than it typically receives.

Future directions include: (1) using a learned rather than templated rationale for the GenRM; (2) integrating the GenRM score as a dense reward signal inside RLOO training; and (3) evaluating curriculum ordering on tasks with genuinely sparse early rewards, where the mechanism we describe here would be expected to have a real effect.

9 Team Contributions

Work was split evenly, with minor role evolution compared to the original proposal:

- **Rakshit Kaushik**: SFT and IPO training loops; fixing the response-token mask bug that affected both IPO and RLOO training; Modal and W&B infrastructure; verifier dataset oracle-labelling pipeline; final report and poster.
- **Rydham Goyal**: RLOO update worker (per-token log-probs, leave-one-out baseline, importance weighting, entropy and KL regularization); GenRM training and inference-time reranker; curriculum RLOO trainer across all four conditions; ablation studies and evaluation pipeline; final report and poster.

Relative to the proposal, curriculum learning replaced the originally planned stretch goal of using GenRM as a dense reward inside RLOO. This change was made because the over-commitment effect we observed in training provided a clearer and more empirically grounded motivation for the curriculum experiment.

References

[1] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.14740.

[2] X. Chen, J. Lu, M. Kim, D. Zhang, J. Tang, A. Piché, N. Gontier, Y. Bengio, and E. Kamalloo. Self-evolving curriculum for LLM reasoning. *arXiv preprint arXiv:2505.14970*, 2025.

[3] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[4] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[5] K. Gandhi, D. Lee, G. Grand, M. Liu, W. Cheng, A. Sharma, and N. D. Goodman. Stream of search (SoS): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

[6] M. Gheshlaghi Azar, M. Rowland, B. Piot, D. Guo, D. Calandriello, M. Valko, and R. Munos. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

[7] S. Parashar et al. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. *arXiv preprint arXiv:2506.06632*, 2025.

[8] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. 2017.

[10] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[11] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[12] L. Zhang, H. Bansal, M. Kazemi, and R. Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.