Extended Abstract

Motivation REINFORCE-style RL fine-tuning methods such as RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024) replace PPO's learned value baseline with a leave-one-out average over K rollouts per prompt. This works well in expectation but suffers from a *zero-gradient pathology*: when all K rewards in a group are equal, every advantage is zero and the group contributes nothing to the gradient. With binary verifier rewards and $K = 4$, we measure this rate climbing from $\sim 22\%$ at the start of training to $\sim 60\%$ by step 100 on Countdown — the second half of training wastes over half its sampling compute on prompts producing no learning signal.

Method We introduce two extensions of RLOO that target this pathology. **Active-RLOO** adds an online filter that drops zero-advantage groups before the gradient step and rescales the loss by the surviving batch size B_{eff}. We show this rescaling is observationally equivalent to running uniform RLOO under an implicit data-adaptive learning-rate schedule pathology. $\eta_{\text{eff}}(t) = \eta/(1 - f_{\text{zero}}(t))$, where $f_{\text{zero}}(t)$ is the zero-gradient fraction at step t. **Adaptive K** is a curriculum-aware controller that one-way ramps rollout count K from 4 to 8 once a rolling-window average of f_{zero} exceeds a threshold (τ=0.5, W=5), supplying more rollouts exactly when they break ties on hard prompts. Both mechanisms are implemented as a small subclass of the project's existing RLOO trainer; all other components (sampling worker, evaluation hooks, KL penalty, checkpointing) are inherited unchanged so curves are directly comparable across runs.

Implementation. Active-RLOO is implemented as a Python subclass of the project's existing RLOO trainer: we override only the training dataset (band-filtered by Phase 1 pass-rate), the update worker (to support a linear-increase LR schedule for ablation studies), and three hooks for Adaptive K's per-step rollout-count control. All other components — sampling worker, in-loop evaluation, KL penalty, checkpointing, wandb logging — are inherited unchanged, which is what makes Uniform-vs-Active comparison curves directly commensurable. The trainer runs on a single H100 via Modal, with the sampling worker (vLLM) and update worker (HuggingFace Transformers, bfloat16 + ZeRO-1) as separate Ray actors that synchronize weights via NCCL after each optimizer step.

Results On Countdown deployment-temperature pass@k (T=0.6, N=16 rollouts), Active-RLOO + Adaptive K achieves the best pass@1 in our sweep (0.586 vs. 0.545 for Uniform RLOO and 0.286 for SFT) and the sharpest policy (1.23× pass@16/pass@1 ratio). Plain Active-RLOO sits between, and the constant 2×10^{-5} LR baseline fares worst on pass@16 (0.660). However, *every* RLOO-family run we test — including Adaptive K — regresses on pass@16 relative to SFT (0.780). This is consistent with the alignment-tax Pareto observed in RLHF (Gao et al., 2022; Kirk et al., 2024): verifier-based RL with a pass@1-style objective sharpens the policy at the cost of sample diversity.

Discussion. On in-loop reward, online filtering is partially substitutable for an LR schedule; on the KL/diversity dimension that ultimately determines pass@k, it is not. The curriculum effect of *which* prompts contribute to each gradient step is structurally distinct from the LR effect of *how strongly* they contribute. That Adaptive K sharpens the policy without de-collapsing it suggests diversity collapse is fundamentally tied to the mean-reward training objective rather than to sample efficiency. Our results come from single-seed runs at 0.5B parameters on one task, so the specific effect sizes should be read as point estimates rather than confidence-bounded claims.

Conclusion. Active-RLOO and Adaptive K give modest pass@1 gains over uniform RLOO with one-line algorithmic changes, and our LR-equivalence ablation isolates a curriculum effect that hand-tuned LR schedules cannot reproduce. The headline open problem is diversity collapse: pass@k-aware update rules (e.g., training on $\max_k$ reward rather than mean reward) and multi-seed/multi-scale validation are the natural next directions.

Active RLOO: Online Filtering with Adaptive K

Zhengmao Liu
Department of Computer Science
Stanford University
zliu1019@stanford.edu

Abstract

REINFORCE-style RL fine-tuning methods like RLOO suffer from a *zero-gradient pathology*: when all K rewards in a leave-one-out group are equal, the group contributes nothing to the update. On Countdown with $K{=}4$ this rate climbs from $\sim$ 22% to $\sim$ 60% over 100 training steps. We introduce **Active-RLOO**, which filters zero-advantage groups before the gradient step and is provably equivalent to running uniform RLOO under an implicit data-adaptive learning-rate schedule, and **Adaptive-K**, which raises rollout count in response to a rolling zero-gradient threshold. An LR-equivalence ablation shows the implicit-schedule view is incomplete: a hand-tuned LR ramp recovers ∼70% of Active-RLOO's in-loop reward gain but fails on the diversity axis — filtering rescales gradient magnitude (LR-like) *and* constrains gradient direction (curriculum-like), and only the latter bounds KL. Adaptive-K achieves our best pass@1 (0.586 vs. 0.545 baseline) but, like every RLOO-family run we test, regresses on pass@16 relative to SFT, leaving diversity collapse as the open problem.

1 Introduction

REINFORCE-style RL fine-tuning methods such as RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024) have become standard for aligning language models to verifiable rewards, replacing PPO's learned value baseline with a group-of-rollouts baseline. A practical limitation of this family is the *zero-gradient condition*: when all K rewards in a group are equal, every leave-one-out advantage is zero and the group contributes nothing to the gradient. With binary verifier rewards and $K = 4$, we measure this rate climbing from $\sim 22\%$ at step 0 to $\sim 60\%$ by step 100 on Countdown — over half the sampling compute in late training produces no learning signal.

Contributions. We introduce two extensions to RLOO addressing this pathology and analyze them jointly:

1. **Active-RLOO**: an online filter that drops zero-advantage groups before the gradient step and rescales by the surviving batch size. We show this is equivalent to running uniform RLOO under an implicit data-adaptive LR schedule $\eta_{\text{eff}}(t) = \eta/(1 - f_{\text{zero}}(t))$.
2. **Adaptive-K**: a controller that bumps K from 4 to 8 once the rolling zero-gradient fraction exceeds a threshold, supplying more rollouts exactly when they break ties on hard prompts.

We then design an **LR-equivalence ablation** testing whether hand-tuned LR schedules on uniform RLOO reproduce Active-RLOO's gains. The answer is nuanced: a linear LR ramp captures ∼70% of in-loop reward gain but fails on the KL/diversity axis — a constant high-LR baseline collapses on pass@16 despite competitive in-loop reward. This separates two mechanisms the LR framing conflates: filtering rescales gradient *magnitude* (LR-like) and constrains gradient *direction* (curriculum-like), and only the latter protects diversity. Adaptive-K achieves our best pass@1 (0.586 vs. 0.545

baseline) but, like every RLOO-family run we test, regresses on pass@16 relative to SFT — diversity collapse remains unresolved.

2 Related Work

REINFORCE-style RLHF. Our work builds directly on RLOO (Ahmadian et al., 2024), which revived value-baseline-free REINFORCE for LLM alignment, and is closely related to GRPO (Shao et al., 2024) which uses an analogous group-relative advantage. Both methods inherit the zero-gradient pathology we characterize; concurrent work on prompt-level filtering (Yu et al., 2025) addresses a similar pathology in PPO contexts, but does not analyze the LR-equivalence we explore here.

Curriculum and active learning for RL. Adapting the training distribution to a model's current capability is a long tradition in RL, from automatic curriculum (Graves et al., 2017; Florensa et al., 2018) to prioritized experience replay (Schaul et al., 2016). Recent work in RLHF has begun to revisit these ideas in the LM setting (Zelikman et al., 2022; Sun et al., 2024), typically by filtering prompts or weighting them by difficulty. Active-RLOO is in this tradition but distinguishes itself by the *minimality* of the intervention (a one-line keep-mask) and the explicit LR-equivalence analysis that connects data-curriculum to optimizer dynamics.

Diversity collapse and the alignment tax. The pattern of pass@1 improvement at the cost of pass@k for larger k is well-documented in RLHF, often framed as an alignment tax or overoptimization effect (Gao et al., 2022; Kirk et al., 2024). Our observation that constant-high-LR runs show runaway KL while Active-RLOO bounds it offers a complementary perspective. Our negative result — that Adaptive-K improves pass@1 but does not recover pass@16 — suggests that diversity collapse is fundamentally tied to the mean-reward objective, motivating pass@k-aware update rules as future work.

3 Method

3.1 Background: RLOO and the Zero-Gradient Condition

Given a prompt x, RLOO (Ahmadian et al., 2024) samples K rollouts $\{y_i\}_{i=1}^{K} \sim \mu(\cdot \mid x)$ from a behavior policy μ and assigns each rollout a scalar reward $r_i \in \{0, 1\}$ from a binary verifier. The leave-one-out advantage uses the mean of the other $K - 1$ rewards as a baseline:

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

The per-step update is the importance-weighted REINFORCE gradient over the $B \times K$ token sequences in the batch:

$$\mathcal{L}_{\text{PG}} = -\frac{1}{BK} \sum_{b,i} w_{b,i} \cdot A_{b,i} \cdot \log \pi_\theta(y_{b,i} \mid x_b), \tag{2}$$

where $w_{b,i} = \text{clip}\big(\exp(\log \pi_\theta - \log \mu),\, e^{-2.3}, e^{2.3}\big)$ is a clipped importance ratio that corrects for off-policy drift between the sampling and update workers. Advantages $A_{b,i}$ are treated as constants w.r.t. θ (`detach()` in code).

Zero-gradient condition. Equation 1 has a degenerate case: when all K rewards in a group are equal, every $A_i \equiv 0$ and that group contributes nothing to the gradient. With $K = 4$ and a binary verifier, this happens for every prompt where the policy either solves all 4 rollouts or fails all 4. We measured this on uniform RLOO at lr $= 10^{-5}$: the rolling zero-gradient fraction climbs from $\sim 22\%$ at step 0 to $\sim 60\%$ by step 100 — over half the sampling compute produces no learning signal in the second half of training.

The base RLOO trainer used as our *Uniform RLOO* baseline (and as the parent class of Active-RLOO) is from the project's earlier phase; this report contributes the filtering, LR-equivalence, and Adaptive-K extensions on top of it.

3.2 Active-RLOO: Online Filtering

Active-RLOO modifies the gradient update to skip zero-advantage groups before the optimizer step. Let $\mathbb{1}_b^{\text{keep}} = \mathbb{1}\{r_{b,1}, \ldots, r_{b,K} \text{ are not all equal}\}$. The filtered loss is

$$\mathcal{L}_{\text{Active}} = -\frac{1}{B_{\text{eff}} \cdot K} \sum_{b,i} \mathbb{1}_b^{\text{keep}} \cdot w_{b,i} \cdot A_{b,i} \cdot \log \pi_\theta(y_{b,i} \mid x_b), \tag{3}$$

where $B_{\text{eff}} = \sum_b \mathbb{1}_b^{\text{keep}}$ is the surviving batch size. The normalization by B_{eff} (rather than B) is what distinguishes Active-RLOO from a no-op: it rescales the gradient to compensate for the dropped groups, so each surviving prompt contributes the same per-token weight as it would in a uniform batch of size B_{eff}.

Filter point. The keep-mask is computed from rewards alone, so we apply it *after* sampling but *before* the gradient computation. This means filtered groups still consume sampling FLOPs but no update FLOPs a deliberate choice, since the sampling cost is fixed by the rollout budget, but the update cost depends on the surviving batch size.

3.3 Implicit Data-Adaptive Learning Rate

A subtle but important consequence of Equation 3: when only B_{eff} of B groups survive, the gradient norm is rescaled by B/B_{eff} relative to uniform RLOO. Because the optimizer step is $\theta \leftarrow \theta - \eta \nabla \mathcal{L}$, this is observationally equivalent to running uniform RLOO at an inflated learning rate

$$\eta_{\text{eff}}(t) = \eta \cdot \frac{B}{B_{\text{eff}}(t)} = \frac{\eta}{1 - f_{\text{zero}}(t)}, \tag{4}$$

where $f_{\text{zero}}(t)$ is the zero-gradient fraction at step t. Since f_{zero} grows monotonically with training (easy prompts saturate first), Active-RLOO's η_{eff} traces an *ascending* schedule that is fully data-driven no manual tuning required. We test in ğ7 whether this LR-schedule perspective alone explains Active-RLOO's gains, by replacing the filter with hand-tuned LR ramps.

3.4 Adaptive K: Curriculum-Aware Rollout Count

A second mechanism uses the same f_{zero} signal to control rollout count K instead of (or in addition to) filtering. The intuition: a high zero-gradient rate means K rollouts are no longer enough to distinguish good prompts from bad increasing K at exactly that point gives more chances for at least one rollout to break the tie.

Let $\bar{f}_t = \frac{1}{W} \sum_{s=t-W+1}^{t} f_{\text{zero}}(s)$ be a rolling average over the last $W = 5$ steps. The Adaptive-K rule is one-way: once the threshold is crossed, K stays at K_{high} for the remainder of training:

$$K_{t+1} = \begin{cases} K_{\text{high}} & \text{if } \bar{f}_t > \tau \text{ or } K_t = K_{\text{high}}, \\ K_{\text{base}} & \text{otherwise.} \end{cases} \tag{5}$$

We use $K_{\text{base}} = 4$, $K_{\text{high}} = 8$, $\tau = 0.5$. The one-way design avoids oscillation: bumping K *causes* f_{zero} to drop (more rollouts $\Rightarrow$ more chances for reward variance), so a hysteresis-free two-way rule would create a feedback loop that ramps K down again. Adaptive-K composes with online filtering and is applied after the filter step.

3.5 Algorithm

Algorithm 1 summarizes one training step of Active-RLOO with Adaptive K. The base RLOO algorithm corresponds to setting $\mathbb{1}^{\text{keep}} \equiv 1$ and $K \equiv K_{\text{base}}$ throughout.

4 Experiment Setup

4.1 Phase 1: Pass-Rate Profiling

Before training, we run a one-shot *pass-rate profile* on all 4,096 training prompts: sample $K_{\text{prof}} = 16$ rollouts per prompt from π_0 at $T = 1.0$, score each with the verifier, and record the empirical pass

Algorithm 1 Active-RLOO with Adaptive K (one optimizer step)

Input: prompts $\{x_b\}_{b=1}^{B}$, behavior policy μ, current policy π_θ, rollout count K_t, threshold τ, window W
Sample: for each b, draw $\{y_{b,i}\}_{i=1}^{K_t} \sim \mu(\cdot \mid x_b)$
Score: compute rewards $r_{b,i} \in \{0, 1\}$ via binary verifier
Filter: $\mathbb{1}_b^{\text{keep}} \leftarrow \mathbb{1}\{r_{b,\cdot} \text{ not all equal}\}$; $B_{\text{eff}} \leftarrow \sum_b \mathbb{1}_b^{\text{keep}}$
Advantages: $A_{b,i} \leftarrow r_{b,i} - \frac{1}{K_t - 1} \sum_{j \neq i} r_{b,j}$
Importance weights: $w_{b,i} \leftarrow \text{clip}\big(\exp(\log \pi_\theta - \log \mu),\, e^{-2.3}, e^{2.3}\big)$
Loss: $\mathcal{L} \leftarrow -\frac{1}{B_{\text{eff}} K_t} \sum_{b,i} \mathbb{1}_b^{\text{keep}} \cdot w_{b,i} \cdot A_{b,i} \cdot \log \pi_\theta(y_{b,i} \mid x_b) + \beta_{\text{KL}}\, \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$
Update: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
Adaptive-K hook: $f_t \leftarrow 1 - B_{\text{eff}}/B$; append to history
  $\bar{f}_t \leftarrow \frac{1}{W} \sum_{s=t-W+1}^{t} f_s$
  if $\bar{f}_t > \tau$ **and** $K_t < K_{\text{high}}$ **then** $K_{t+1} \leftarrow K_{\text{high}}$ **else** $K_{t+1} \leftarrow K_t$

rate $\hat{p}(x) \in \{0, 1/16, 2/16, \ldots, 1\}$. The profile is dumped to JSONL and consumed by Active-RLOO's dataset wrapper at training time.

Filtering band. By default we use the soft band $\hat{p} \in [10^{-9}, 1-10^{-9}]$, which removes only the "Dead" ($\hat{p} = 0$, all 16 SFT rollouts failed) and "Solved" ($\hat{p} = 1$, all 16 succeeded) tiers, and keep all unprofiled prompts. We deliberately avoid tighter bands (e.g., $\hat{p} \in [0.2, 0.8]$) for two reasons. First, the profile is a step-0 snapshot; pass-rates drift as training progresses (easy prompts saturate, hard ones become solvable), so a tight static band over-commits to a stale signal. Second, our Phase 1 study found only a weak correlation between mean per-token entropy and pass-rate ($\rho \approx -0.14$), undermining confidence in fine-grained tier assignments. The soft band's role is therefore narrow it removes only prompts that are mathematically guaranteed to produce zero advantage at $K = 4$ given that they already did at $K = 16$ and we delegate the dynamic decision to the online filter (§3.2), which acts on current-step rewards. Profiling uses the same vLLM worker setup as training.

4.2 Training Configuration

Table 1 lists all training hyparameters in addition to the common settings from SFT/IPO. The base configuration is shared across all RLOO/Active-RLOO runs; per-run differences are confined to learning-rate schedule and Adaptive-K toggle.

Table 1: Hyparameters in addition to the SFT/IPO settings. Per-run overrides in Table 2.

Parameter	Value
Rollouts per prompt K (base)	4
Importance-ratio clip range	$[e^{-2.3}, e^{2.3}]$
Zero gradient ratio threshold	0.5

4.3 Run Inventory

Table 2 lists the five RLOO-family runs that comprise the main comparison. Runs differ only along three axes: (i) whether the online filter is enabled, (ii) the learning-rate schedule, and (iii) whether Adaptive-K is enabled.

The two LR-control runs (constant 2×10^{-5} and linear ramp $1.3 \to 3.4 \times 10^{-5}$) are calibrated to match Active-RLOO's measured η_{eff} trajectory from Equation 4: the constant matches its time-averaged effective LR, and the linear ramp matches its endpoints. Together they probe whether Active-RLOO's gains over uniform 10^{-5} can be reproduced by hand-tuned LR alone.

4.4 Evaluation

We use two complementary evaluation modes:

Table 2: Run inventory. All runs share Table 1; differences below.

Run	Filter	LR schedule	K	Purpose
Uniform RLOO (10^{-5})	—	constant 10^{-5}	4	baseline
Uniform RLOO (2×10^{-5})	—	constant 2×10^{-5}	4	avg-LR control
Uniform RLOO (linear ramp)	—	$1.3 \to 3.4 \times 10^{-5}$	4	LR-trajectory control
Active-RLOO	✓	constant 10^{-5}	4	main method
Active-RLOO + Adaptive-K	✓	constant 10^{-5}	$4 \to 8$	curriculum-aware

In-loop evaluation. Every 5 optimizer steps, we evaluate on the held-out test set with $N_{\text{eval}} = 16$ rollouts per prompt at the *training* temperature ($T = 1.0$). We log: `test/rollout_accuracy` (mean reward over the 16 rollouts), `test/pg_loss`, `test/kl_loss`, and `test/importance_weight_mean`. This mode runs on the same vLLM worker used for training and adds negligible wall-clock overhead. We added these metrics on top of the base RLOO trainer for this project; the IPO trainer from the prior project phase was instrumented analogously.

Offline pass@k evaluation. At the end of training, we evaluate each final checkpoint on the full test set under *deployment* sampling parameters: $T = 0.6$, top-p=0.95, top-k=20, $N = 16$ rollouts per prompt. From these 16 rollouts we report `pass@1` (mean per-rollout accuracy), `pass@16` (probability at least one rollout is correct), and the *diversity ratio* `pass@16`/`pass@1` as a single-number proxy for sample diversity (higher = more diverse). The temperature mismatch between training ($T = 1.0$) and deployment ($T = 0.6$) is deliberate: it surfaces diversity collapse that is invisible at training temperature. This offline evaluation reuses the existing Modal-based evaluation harness from the project's earlier SFT, IPO, and RLOO phases — the same harness, sampling parameters, and metric definitions used to compare those checkpoints — so Active-RLOO's pass@k numbers are directly commensurable with the project's prior baselines.

5 Results

We organize results around three questions: **(R1)** does online filtering improve test-time reward in-loop? **(R2)** is the in-loop gain explained by an implicit LR schedule? **(R3)** how do these gains translate to deployment-temperature pass@k, and can Adaptive-K mitigate the trade-off?

5.1 Headline: Deployment-Temperature Pass@k

Table 3 reports offline pass@k on the full Countdown test set under deployment sampling parameters from the SFT/IPO settings. All RL runs improve pass@1 substantially over SFT, but most *regress* on pass@16 — the canonical signature of diversity collapse (Gao et al., 2022). Active-RLOO + Adaptive-K achieves the best pass@1 (0.586) and the sharpest model (lowest pass@16/pass@1 ratio, 1.23×); plain Active-RLOO sits between Uniform RLOO and Adaptive-K on both axes.

Table 3: Offline pass@k at deployment sampling. **Bold** = best per column. Diversity ratio = pass@16 / pass@1; lower means a sharper, less diverse policy.

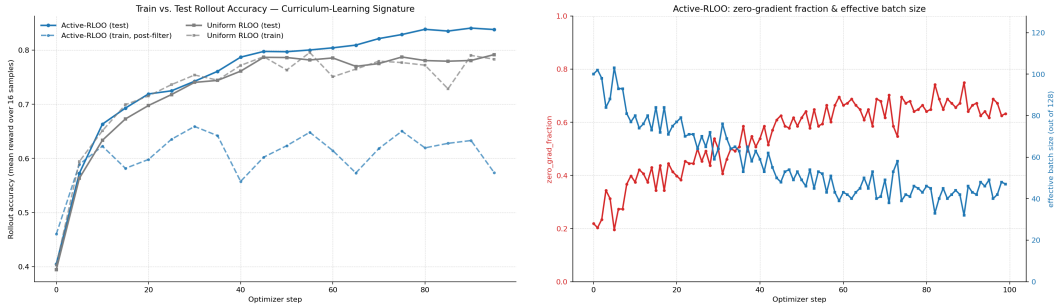
Run	pass@1	pass@16	Ratio
SFT (base policy)	0.286	**0.780**	2.73×
Uniform RLOO (10^{-5})	0.545	0.760	1.39×
Uniform RLOO (2×10^{-5}, const.)	0.506	0.660	1.30×
Uniform RLOO (linear ramp)	0.520	0.720	1.38×
Active-RLOO (online filter)	0.526	0.700	1.33×
Active-RLOO + Adaptive-K	**0.586**	0.720	**1.23**×

5.2 R1: In-Loop Gain from Online Filtering

Figure 1a overlays train and test rollout accuracy (at training $T = 1.0$) for Uniform RLOO and Active-RLOO across all 250 optimizer steps. Two patterns are evident:

1. **Active-RLOO's test curve climbs faster** than Uniform's, reaching $\sim$0.85 by step 100 vs. $\sim$0.75 for Uniform — a 10-point gap that persists through the rest of training.
2. **Active-RLOO's train curve crosses below its test curve** around step 60 and stays there. This is the canonical *curriculum-learning signature*: easy prompts saturate ($\hat{p} \to 1$ under the current policy), drop into the all-pass tier, and get filtered out — so the surviving training pool gets harder over time even though the test set is fixed. Uniform RLOO's train and test curves track each other throughout, as expected.

Figure 1b confirms the filtering mechanism on twin axes: the zero-gradient fraction climbs from ~ 0.22 at step 0 to ~ 0.60 by step 100, meaning the effective batch size shrinks from 128 to 50 over the same window. The optimizer step continues to run on the surviving groups; that's what powers the in-loop test gain.



(a) Train and test rollout accuracy. Active-RLOO's train accuracy crosses *below* its test accuracy — the curriculum-learning signature.

(b) Zero-gradient fraction (red, left axis) and effective batch size (blue, right axis). Effective batch shrinks from 128 to $\sim$56 as easy prompts saturate.

Figure 1: In-loop diagnostics for Active-RLOO vs. Uniform RLOO. Left: the curriculum-learning signature in train/test accuracy. Right: the filtering mechanism that produces it.

5.3 R2: LR-Equivalence Ablation

A natural alternative explanation for R1 is that online filtering is "just" an implicit LR schedule (Equation 4). If true, hand-tuned LR schedules on Uniform RLOO should reproduce the gain. We test this with two Uniform-RLOO variants calibrated to Active-RLOO's measured η_{eff} trajectory: a constant $\text{lr} = 2 \times 10^{-5}$ (matches the time-averaged effective LR) and a linear ramp $1.3 \times 10^{-5} \to 3.4 \times 10^{-5}$ (matches the endpoints).

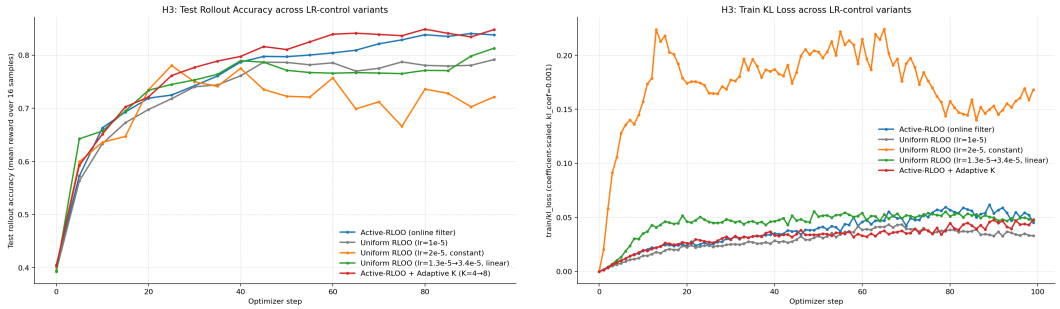
Figure 2a compares all four runs' in-loop test accuracy. The linear ramp recovers $\sim$70% of Active-RLOO's gain over Uniform 10^{-5}; the constant 2×10^{-5} recovers less. *In-loop, the LR-equivalence hypothesis is partially supported*: filtering can be largely mimicked by a well-chosen ramp.

But the LR-equivalence story breaks down at deployment temperature. The constant 2×10^{-5} run — which has *higher* training gradient norms than Active-RLOO — collapses on pass@16 (0.660, the worst of any RLOO run). Figure 2b shows why: its train KL trajectory runs away to ~ 156 nats by end of training, whereas Active-RLOO and the linear ramp keep KL bounded below ~ 70 nats. *LR equivalence holds for in-loop reward but not for the KL/diversity dimension — which is what ultimately determines pass@16.*

5.4 R3: Adaptive-K and the Diversity Trade-off

The pass@k table reveals an unexpected pattern: *every RLOO-family run regresses on pass@16*. This is consistent with the "alignment-tax"-style Pareto observed in RLHF (Ouyang et al., 2022) — RL fine-tuning sharpens the policy at the expense of sample diversity, and the verifier reward used during training (pass@1 at $T = 1.0$) gives no incentive to preserve diversity.

Adaptive-K is our Phase-3 attempt to mitigate this. The intuition is that *when the zero-gradient fraction is high, K is the bottleneck* — not the LR — so we increase K in response. Figure 3 shows the mechanism: zero-gradient fraction crosses the τ=0.5 threshold around step 24, triggering the



(a) In-loop test rollout accuracy. The linear ramp recovers most of Active-RLOO's gain over Uniform 10^{-5}.

(b) Train KL loss. Constant lr $= 2 \times 10^{-5}$ shows runaway KL, while Active-RLOO and the linear ramp stay bounded.

Figure 2: H3 LR-equivalence ablation across four LR-control variants. In-loop reward (left) is approximately LR-equivalent — a hand-tuned linear ramp on Uniform RLOO recovers most of Active-RLOO's gain. KL/diversity (right) is *not*: constant lr $= 2 \times 10^{-5}$ matches Active-RLOO's average effective LR but its KL diverges, foreshadowing its pass@16 collapse in Table 3.

one-way ramp from $K{=}4$ to $K{=}8$, which immediately drops zero-gradient fraction back below 0.4 (more rollouts $\Rightarrow$ more chances for reward variance).

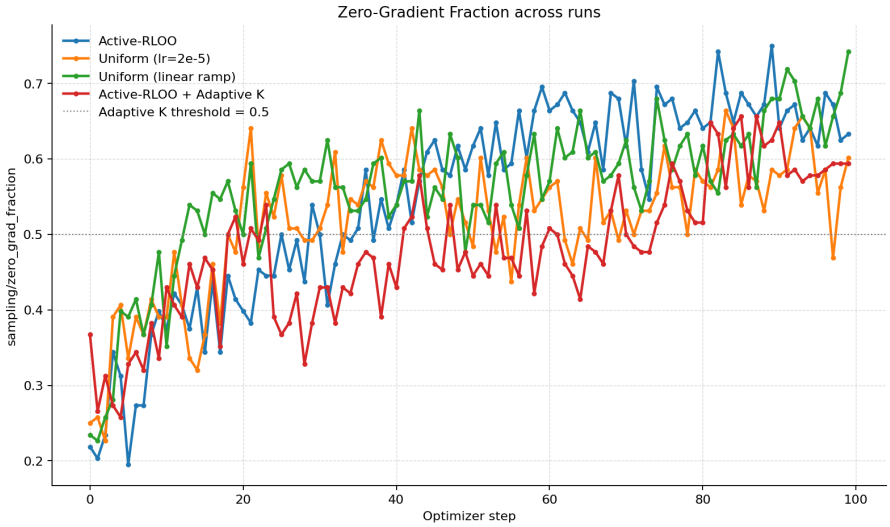


Figure 3: Zero-gradient fraction across runs. Adaptive-K (red) responds to the rolling threshold by bumping K from 4 to 8, producing the visible discontinuity around step 24.

The pass@k result (Table 3) shows Adaptive-K achieves the *best pass@1 of any run (0.586)* and the lowest diversity ratio (1.23×) — i.e., the sharpest policy. It does *not* recover pass@16 above SFT, which suggests that simply adding rollouts is not sufficient to prevent diversity collapse: a separate diversity-aware objective (e.g., entropy regularization tuned at deployment temperature, or a pass@k-aware reward) is likely needed. We discuss this further in §6.

6 Discussion

6.1 Interpreting LR-Equivalence

The R2 ablation gives a more nuanced answer than the binary "equivalence/non-equivalence" framing suggests. Online filtering is *partially* equivalent to a hand-tuned LR schedule: a linear ramp on Uniform RLOO recovers ∼70% of Active-RLOO's in-loop reward gain. This matches the predic-

tion from Equation 4 — if filtering rescales the gradient by $1/(1 - f_{\text{zero}}(t))$, an explicit schedule that traces the same effective-LR trajectory should produce the same reward dynamics.

Where the equivalence *breaks* is on the KL/diversity dimension. The constant $\text{lr} = 2 \times 10^{-5}$ run matches Active-RLOO's time-averaged η_{eff} and achieves competitive in-loop reward, but its train KL diverges and its pass@16 collapses. We attribute this to a mechanism the LR-equivalence story does not capture: filtering changes *which* prompts contribute to each gradient step, not just *how strongly*. By dropping all-pass/all-fail groups, Active-RLOO concentrates each update on prompts near the policy's current decision boundary — a data-curriculum effect that constrains the direction of policy change, not just its magnitude. A hand-tuned LR schedule can match the magnitude, but not this directional constraint.

This refines a common framing in the RLHF literature: data-selection techniques are often discussed as orthogonal to optimization hyperparameters, but our results suggest the two are entangled along the reward axis (where they substitute for each other) and decoupled along the diversity axis (where data selection has a structural effect that LR tuning does not).

6.2 Diversity Collapse as the Unsolved Problem

Every RLOO-family run in Table 3 *regresses on pass@16* relative to SFT. This is consistent with prior observations of an alignment-tax-style Pareto in RLHF (Gao et al., 2022; Kirk et al., 2024): verifier-based RL with a pass@1-style training objective sharpens the policy and depresses sample diversity, even when the policy improves on the objective being optimized.

Adaptive-K achieves the best pass@1 in our sweep (0.586) and the sharpest policy (1.23× diversity ratio) but does *not* mitigate pass@16 collapse. This is empirical evidence that diversity collapse is not primarily a sample-efficiency problem (otherwise more rollouts per prompt would help) but a problem with the *training objective itself*: any update rule that uses mean per-rollout reward as its signal will push the policy toward a single high-reward mode. Mitigating this likely requires a pass@k-aware objective — either explicit (e.g., reward the maximum of k rollouts rather than the mean) or implicit (e.g., entropy regularization tuned at deployment temperature rather than training temperature, which is what our current 0.01 entropy bonus controls).

6.3 Limitations

Several factors limit the strength of our conclusions.

Scale and breadth. All experiments use a single 0.5B-parameter model on a single task (Countdown). The zero-gradient pathology and diversity collapse phenomena are well-documented at larger scales, but the magnitudes of our effect sizes (the 70% LR-equivalence figure, the Adaptive-K pass@1 gain) may not transfer. Multi-scale validation on Qwen-1.5B and a multi-task replication on GSM8K are planned as follow-ups.

Static profile. The Phase 1 pass-rate profile is computed once on the SFT checkpoint and never refreshed. Since pass-rates drift during training, a stale profile means the static band's role shrinks toward zero over time — the online filter ends up doing essentially all of the work. Periodic re-profiling (e.g., every 50 steps) is a natural extension but was out of scope here.

Verifier-only reward. The binary verifier reward has no notion of solution quality (multiple correct expressions all score 1). This amplifies the diversity-collapse problem: nothing in the training signal rewards exploring alternative correct solutions. Results may differ under denser reward signals (e.g., process-reward models or graded verifiers).

6.4 Future Work

The most natural next steps from this report are:

- **Pass@k-aware objectives.** Replace the mean-reward update with a $\max_k$ reward (training the policy to maximize the probability that *at least one* of k rollouts succeeds) to directly target the metric that captures diversity.

- **Deployment-temperature entropy regularization.** Tune the entropy bonus against pass@16 at $T = 0.6$ rather than only observing it via the train-time entropy term.
- **Two-stage Adaptive-K with hysteresis.** The one-way ramp in this report avoids a feedback loop but discards the possibility of ramping K back down once filtering has stabilized.

7 Conclusion

We introduced Active-RLOO, a minimal extension of RLOO that filters zero-advantage groups before the optimizer step, and Adaptive-K, a curriculum-aware mechanism that ramps rollout count K in response to the zero-gradient rate. On Countdown with a 0.5B-parameter model, both mechanisms improve in-loop test reward and deployment-temperature pass@1 over Uniform RLOO; Adaptive-K achieves the best pass@1 of any run in our sweep (0.586 vs. 0.545 baseline).

Our LR-equivalence ablation establishes that online filtering is partially equivalent to an implicit data-adaptive LR schedule for in-loop reward, but *not* for KL/diversity — a hand-tuned LR schedule matching Active-RLOO's effective-LR trajectory still collapses on pass@16. This separates two mechanisms that the LR-schedule framing conflates: filtering rescales gradient magnitude (an LR effect) and constrains gradient direction (a curriculum effect). The curriculum effect is what protects KL/diversity, and it cannot be reproduced by LR tuning alone.

Diversity collapse remains the unresolved problem: every RLOO-family run in our sweep — including Adaptive-K — regresses on pass@16 relative to SFT. Our results suggest this is not a sample-efficiency issue but a limitation of the mean-reward training objective itself, motivating pass@k-aware update rules as the natural next direction.

Changes from Proposal Three findings during the project drove substantive deviations from the proposal. **(i) Entropy → pass-rate.** Phase 1 profiling found only a weak correlation between mean per-token entropy and verifier pass-rate ($\rho \approx -0.14$), so we replaced entropy with pass-rate itself — a direct measure of reward variance — as the curriculum signal. **(ii) Static Goldilocks bin → online filter.** We retained the proposal's static band only as a coarse sanity guard ($\hat{p} \in [10^{-9}, 1-10^{-9}]$) and moved principal filtering to post-rollout, pre-update, acting on *current-step* rewards rather than a step-0 prior. This shift is what makes the filter robust to policy drift and gives rise to the implicit data-adaptive LR of Equation 4. **(iii) Adaptive Thaw → Adaptive K.** With entropy demoted as a signal, the proposal's threshold-shifting "Thaw" scheduler no longer fit; Adaptive K (§3.4) replaces it with a mechanistically simpler controller that bumps rollout count in response to the rolling zero-gradient rate.

In addition, the LR-equivalence ablation (§5.3) was added mid-project as the most non-obvious analytical contribution, and the proposal's OOD 5-digit Countdown evaluation was deprioritized in favor of deeper in-distribution pass@k analysis once diversity collapse emerged as the central finding.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. arXiv:2402.14740 [cs.LG] `https://arxiv.org/abs/2402.14740`

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic Goal Generation for Reinforcement Learning Agents. arXiv:1705.06366 [cs.LG] `https://arxiv.org/abs/1705.06366`

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling Laws for Reward Model Overoptimization. arXiv:2210.10760 [cs.LG] `https://arxiv.org/abs/2210.10760`

Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated Curriculum Learning for Neural Networks. arXiv:1704.03003 [cs.NE] `https://arxiv.org/abs/1704.03003`

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the Effects of RLHF on LLM Generalisation and Diversity. arXiv:2310.06452 [cs.LG] https://arxiv.org/abs/2310.06452

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. arXiv:1511.05952 [cs.LG] https://arxiv.org/abs/1511.05952

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL] https://arxiv.org/abs/2402.03300

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision. *arXiv preprint arXiv:2403.09472* (2024).

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476 [cs.LG] https://arxiv.org/abs/2503.14476

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. arXiv:2203.14465 [cs.LG] https://arxiv.org/abs/2203.14465