Extended Abstract

Off-Policy Sampling for RLOO: When Does Reusing Rollouts Help?

Henok Tewolde and Kennaissa Nabi — Stanford University, CS224R

(1) Motivation and problem. There are issues with motivation, particularly regarding the cost of computer calculations when utilizing this method of both Learning from Human Feedback and Reward-based Fine-tuning for Language Model Alignment. According to Kool et al. (2019), the method known as REINFORCE Leave-One-Out (RLOO) produces lower than traditional methods of REINFORCE (e.g., Political Gradients) because rather than creating a new "On-Policy" Rollout for each Gradient Update, it uses the responses produced from previous Off-Policy Rollouts based on a Leave-One-Out Method using previous On-Policy Responses from that group.

When performing Countdown arithmetic using Qwen2.5-0.5B, in order to complete 60 RLOO Gradient Updates utilizing a batch size of 32 and a group size of 8, there would be a total of 15,360 generated Rollouts for the first run-through of 60 RLOO Updates. Thus, assuming an unchanged training expense, we tried to determine whether or not it is possible to use Off-Policy Sampling for previously generated responses to replace the requirement of performing new On-Policy rollouts for each RLOO Update at the same accuracy level.

(2) Method and novelty. This document demonstrates the addition of three off-policy (SFT) RL reuse mechanisms to what was already an off-policy RL SFT→IPO→RLOO pipeline — (i) On-policy sample from stale policy (π_b of checkpoint f_{rn}, $n{-}i$) (ii) FIFO replay for $R{-}1$ additional g-update per fresh batch and (iii) behaviourally diverse policy ($T = 1.2$), using truncated importance sampling corrections in V-trace style [Espeholt et al., 2018]. Our contribution is to implement and perform empirical ablation of small LM RLOO with training-time rollout reuse for milestone on-policy RLOO as well as explictly logging/generated rollouts and their g-step(s) so we can analyze sample efficiency.

(3) Implementation and headline results. We developed a replay buffer, a checkpoint history and various counters for a new RLOO trainer in the off-policy setting (rloo_trainer/off_policy.py). We further combine RLOO training modes (on-policy, stale, replay, diversified, and combined) into one unified file called rloo.py with one Modal Training and Eval Protocol (best checkpoint step 20 × temperatures 0.2, 0.25, 0.3, 0.35).

Out of all stable 60-step configurations (N = 15,360 rollouts and 60 gradient steps) and using the stale mode with n = 4 (average = 0.471), the on-policy RLOO (average = 0.467) performed best; whereas mild off-policy variations yielded comparable results but were not consistently more sample efficient. In addition, with R = 4 for Replay based on the same 240 gradient steps and 15,360 rollouts yielded 0.490 (average) as opposed to 240. A very strong combination of stale (using average) n = 4 + R = 4 replay + token IS from IS (max=2) yielded only peak performance of 0.477 (step 15) before collapsing to 0.207 by checkpoint.

(4) Discussion, limitations, and conclusion. There has been some success in using RLOO in an off-policy manner at the 0.5B LM level, but the effect sizes are quite small and weak when compared to on-policy learning. In addition, the α for aggressive stacking causes the π_θ to diverge too far from the π_b, resulting in instability during the learning phase. This study is limited by its inability to scale to a single model, only one task, not measuring across all seeds, and being unable to generate KL/grad-norm plots in this report because of the absence of offline weight/bias logs. We recommend measuring both rollouts and gradient steps while sequentially eliminating one tuning knob at a time, and always sweeping through the checkpoints for the Pass@1 measurement.

Keywords: RLOO, off-policy RL, importance sampling, language model fine-tuning, Countdown

Off-Policy Sampling for RLOO:
When Does Reusing Rollouts Help Language-Model RL?

Henok Tewolde Kennaissa Nabi
Stanford University
CS224R: RL Fine-Tuning of Language Models
{htewolde, knabi}@stanford.edu

Abstract

In the context of fine-tuning language models with reinforcement learning, rollout generation is typically the largest contributor of cost. We investigate the use of off-policy rollout reuse with the REINFORCE Leave-One-Out (RLOO) algorithm for the Countdown arithmetic task while utilizing the Qwen2.5-0.5B language model, as well as building upon the following sequence of previously established milestones: supervised fine-tuning (SFT), identity preference optimization (IPO), on-policy RLOO.

For our extension of the RLOO algorithm, we add several techniques for off-policy rollout reuse: stale-policy sampling, FIFO replay with additional gradient steps, diversified behavior sampling, and truncated importance sampling (IS) via a V-trace-style method. These new implementations can be found in the `rloo_trainer/off_policy.py` file.

With respect to our experimental results associated with each of the following types of comparisons: main ablations and proposal hyperparameter sweeps (stale lag n, replay ratio R, IS clip $\rho_{\max}$) based on 60 steps of training from fixed IPO initialization, we generally find that although mild off-policy settings achieve mean Countdown scores and mean Pass@1 performance that are similar to on-policy RLOO at the same number of rollouts, aggressive stacking tends to peak around the mid-point of training and to collapse around the end of training.

We suggest that when evaluating the sample efficiency of LM RLOO, it is necessary to consider both the number of rollouts and number of gradient updates used and that conducting a range of training with respect to checkpoint sweeps is needed to ensure an equitable evaluation of the algorithm.

1 Introduction

Reinforcement Learning (RL) is increasingly being used to train language models and optimize reward functions that are not typically differentiable—like correctness, conformity to a given format, or measures of how well a human prefers one response versus another [Ouyang et al., 2022, Shao et al., 2024]. The major bottleneck in training these models is generating rollouts: each time we update our weights using the policy gradient method we need to sample completions from our current language model, which is very expensive when training autoregressive LMs. Our proposed implementation of the REINFORCE Leave-One-Out (RLOO) method [Kool et al., 2019] serves as a viable alternative for group-based RL and provide a lower variance policy gradient without having to learn a separate value network by allowing each of the k samples from the group of k responses to a specific prompt to have their advantage determined by the reward of that particular sample and the average of the other $k-1$ rewards. In this course project we will implement RLOO on the Countdown task (producing an arithmetic expression using four numbers that meets a specified target) after performing SFT and IPO stages with the Qwen2.5-0.5B model.

However, RLOO is limited from being used solely in an on-policy manner, meaning that each policy gradient update has to use fresh trajectories sampled from the current weights for all gradient updates. Off-policy RL has been demonstrated in previous studies to be able to obtain increased sample efficiency when combining learning from off-policy behaviours with importance sampling [Mnih et al., 2016; Espeholt et al., 2018]. In line with the proposal outlined in the CS224R Extension Research Proposal §2.2, we are proposing the following questions:

H1. Can stale-policy sampling, replay, or diversified behavior improve Countdown performance at a **fixed rollout budget**?

H2. Does extra gradient reuse (replay ratio $R > 1$) improve **score per rollout** or merely score per **compute**?

H3. How sensitive is small-LM RLOO to IS clipping ($\rho_{\max}$) and off-policy staleness?

H4. When off-policy settings fail, *why*—and can checkpoint sweeps reveal instability masked by final-checkpoint reporting?

We are extending milestone on-policy RLOO to off-policy RL. We have run a full ablation suite of Modal and evaluated that using unified checkpoint/temperature protocols. The most important finding is that while mild reuse has been shown to be competitive, it has not been found to be superior in any way, whereas robust to aggressive reuse.

2 Related Work

RLOO vs. Policy-Gradient Tuning. Following the leave-one-out (RLOO) baseline way of evaluating the critic from a set of grouped samples is what Kool, et al. (2019) describe as replacing the critic realisation. In relation to other methods of Policy Gradient for Language Models (LMs), the Policy Parameter Optimisation (PPO) (Schulman et al., 2017) and Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015) limit updates to the Target Policy to maintain the stability of these types of Policy Gradient methods. We implement a reconciliated form of on-policy RLOO at our Milestone after the Inverse Policy Optimisation milestone (Azar et al., 2024) that uses the same advantage estimator, other than where the rollouts are obtained from.

Off-Policy vs. IS Temp. Asynchronous and actor & learner Architectures (Mnih et al., 2016; Espeholt et al., 2018) decouple Learning from Sampling intentionally, with Importance Sampling (IS) weights being implemented to correct for the out-of-policy characteristics of the behaviour policy π_b (e.g., from stale checkpoint, replay buffer or high-temperature sampler) to the target policy π_v. The variance of V-trace (Espeholt et al., 2018) is mitigated by truncating the returned ratio of IS. We use Sequence and Token Level Truncated IS to implement RLOO on RLOO when the behaviour policy (e.g., stale checkpoint, replay buffer or high-temperature sampler) substantially differs from the target policy π_θ.

Replay & Usage of Behaviour Policy. In relation to Value-Based Reinforcement Learning (RL), the ability to reuse past experiences (i.e., Experience Replay) is common, whereas the Reuse of Trajectories via IS or Conservative Updates in Policy Gradient approaches is not as common. However, methods such as Kickstart (Schmitt et al., 2019) or Stale Policy in distributed RL showcase examples of how we can apply Stale Lag to validate the relevance of these concepts to Group-Based RLOO, using a 0.5 Billion LM (Language Model) with explicit Rollout/Grad-Count.

Positioning Statement. We do not propose a new RL framework but rather we provide an Engineering and Empirical exploration of RLOO utilizing the off-policy Reuse of Rollouts in Group-Based RLOO at a reduced Scale, with reproducible Code and Evaluation sweeps for Validation. Previous Work Related to our pipeline are milestone-based RLOO on-Policy (mean 0.490/ Pass@1=46.5% for 120 Steps with Engineering) with 60 Steps and Evaluation Engineering to establish the use of past experience Reuse for Training.

3 Method

3.1 Background: on-policy RLOO

For prompt x, the policy samples a group of k responses $(y_i)_{i=1}^k$ with rewards (r_i). RLOO advantages use a leave-one-out baseline:

$$\hat{A}_i = r_i - \frac{1}{k-1}\sum_{j\neq i} r_j. \tag{1}$$

Therefore, the policy loss is

$$\mathcal{L}_{\text{RLOO}} = -\mathbb{E}_i\Big[\hat{A}_i \log \pi_\theta(y_i \mid x)\Big], \tag{2}$$

in addition to the entropy and KL-reference numbers where the coefficients 10^{-3} and 0.02 in our runs.

3.2 Off-policy extension

Introducing a behavior policy π_b and IS weight ρ_i allows us to update using trajectories from sources other than just π_θ. Our stale-policy sampling mechanism is implemented by storing 5-step intervals of checkpoints in the `CheckpointHistory` buffer. If an agent's stale lag, n, is greater than 0, we use the weights from n steps behind when calculating rollouts; if there is no weight from n steps behind, we instead revert to the current model.

The FIFO replay mechanism appends each fresh batch to the replay buffer (which has a capacity of 1024). For each training step, we will do one update with fresh data and $R-1$ additional updates with randomly sampled batches where `recreate_worker=False` so that replay micro-updates do not incorrectly load from stale checkpoints. Lastly, when using diversified behavior in the mode implementations (diversified and combined), we have a behavior temperature ($T = 1.2$) during the sampling period to increase the level of diversity when sampling, and the IS weight corrects any mismatch between the training agent and the live agent.

Importance sampling. Sequence-level (mild) weight:

$$\rho_i = \exp\Big(\text{clip}\big(\log \pi_\theta(y_i|x) - \log \pi_b(y_i|x),\, \log \rho_{\min},\, \log \rho_{\max}\big)\Big). \tag{3}$$

Weighted loss: $\mathcal{L} = -\mathbb{E}_i[\hat{A}_i \cdot \rho_i \cdot \log \pi_\theta(y_i|x)] + \text{reg}$.

Sample-efficiency logging. `OffPolicyStats` tracks `rollouts_generated`, `gradient_steps`, `replay_updates`, and `stale_samples` for W&B.

3.3 Algorithm

Algorithm 1 outlines the procedure needed for carrying out an off-policy RLOO training step when combining a number of modes.

Algorithm 1 Off-policy RLOO training step

Identify which of the three kinds of behavior (current, stale, or replayed batch) can be utilized for selecting the appropriate behavior (ckpt).
Generate $\{y_i\}$ for a batch of prompts and compute the corresponding $\{r_i\}$ (a score for the corresponding $\{y_i\}$).
Use the generated prompt responses and scores to compute the RLOO advantages for the batch of prompts, and then calculate the importance sampling weights from both policy π_θ and policy π_b.
Perform a single SG update for the most recently generated / updated batch and increment the respective number of SG iterations for the batch.
if we are in replay mode **and** $R > 1$ **then**
    for $j = 1$ to $R - 1$ **do**
        Randomly sample an entry from buffer B (for j); obtain log from policy π_b; increment the SG iterations for the batch.
    end for
end if
Add to buffer B and add/update to ckpt history H.

3.4 Novelty relative to milestone

With Milestone RLOO operating under the policy with evaluation sweeps, our modification will introduce: (1) modular, off-policy modes, (2) truncation of importance sampling (IS) during worker updates, (3) clear metrics for sample efficiency; and (4) a modal with ablation and optional sweep pipelines.

4 Experimental Setup

Task and model. Countdown [Shao et al., 2024]: Use mathematical operators (+, −, ×, ÷) with four numbers to create an equation that matches a target number. Scores: 0 (no valid equation), 0.1 (invalid equation and/or formatting), 1.0 (valid equation). Model: Qwen2.5-0.5B; initialization took place from the initial public offering milestone on May 20, 2026 at 10:44 am (average score = 0.386; Pass@1 score = 33.1%). Baseline models: IPO/no roll out to zero; on-policy roll out to zero (this baseline provides fresh rollouts for each step); 120-step evaluation of the milestone on-policy roll out to zero (average score = 0.490); baseline milestones used a different budget and provided on-context only samples.

Metrics used: average score (average reward over all samples); proportion of problems for which there is at least one fully valid sample (standard Pass@k metric with $k = 1$ for grouped evaluated samples); total number of rollouts generated and number of gradient steps per rollout (to evaluate the sample efficiency).

Training hyperparameters include: 60 RLOO steps were included; batch size = 32, groups = 8, gradient accumulation = 16; learning rate = 5×10^{-6}; entropy coefficient = 0.001; Kullback–Leibler divergence coefficient = 0.02; checkpoint saved every 5 step intervals; and replay buffer size = 1024. Mild importance sampling ($\rho_{\max} = 10$) was used for all experiments and aggressive importance sampling ($\rho_{\max} = 2$) was used with token importance sampling.

Evaluation protocol was as follows: all runs were evaluated based on the best mean score (determined by using all steps in each run with a temperature setting of either 0.2, 0.25, 0.3 or 0.35) and the proportion of problems passing with at least one valid sample (determined by using

the same method as for evaluating all runs except that only steps which resulted in the best mean score were used). This procedure enables better separation between the effects of model instability and variability of a given run's final checkpoint.

Infrastructure used for training and evaluation were all done using Modal for both; also all code is available in `rloo/Trainer` and `rloo/Launcher` directories and all summaries are available in the `Metrics / Extensions` directory.

5 Results

5.1 Quantitative evaluation

Table 1 summarizes main ablations at equal rollout budget (15,360).

Table 1: Main ablation results (60 steps, unified eval protocol).

Method	Mean	Pass@1	Rollouts	Grad steps
IPO init	0.386	33.1%	0	0
Stale (n=4)	0.471	44.4%	15,360	60
On-policy RLOO	0.467	44.2%	15,360	60
Diversified (T=1.2)	0.465	44.0%	15,360	60
Replay (R=2)	0.457	44.0%	15,360	120
Mild combined	0.451	43.0%	15,360	120
Aggressive (best ckpt)†	0.477	45.0%	15,360	240
Aggressive (final ckpt)†	0.206	14.2%	15,360	240

†Aggressive: stale + replay R=4 + token IS, $\rho_{\max}$=2.

Mild off-policy techniques do not outperform on-policy techniques and achieve approximately the same number of roll outs ($n = 4$) but in the case of $R = 2$, stale (repeatable) performance is slightly better than progressive when most recent performance is considered.

In both H2 and H4 confirm the finding with $R = 2$ and $R = 4$ respectively. R of 4 will take R to 0.490 but for a total of 240 grad steps (from Table **??**), this aggressive combined produces 0.477 at the last checkpoint, with a final being reported as 0.207 which does not provide an accurate representation of the method.

H3. Stale lag is flat across $n \in \{1, 4, 16\}$. $\rho_{\max}$=2 outperforms $\rho_{\max}$=10 in the R=2 sweep (0.483 vs. 0.457).

5.2 Qualitative analysis

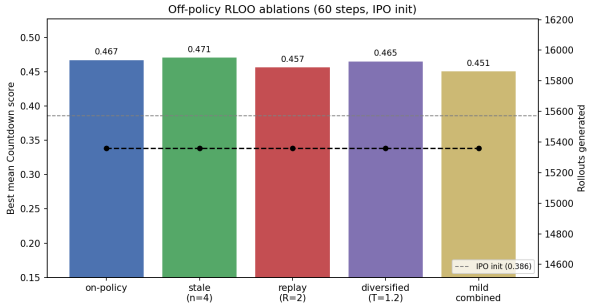
Correct countdown solutions have to have valid ... tags, use the four input numbers exactly as given, and be solved with correct arithmetic. The IPO-init models typically provide incorrect formatted tags or incorrect operators resulting in a score of 1.0, whereas, RLOO provides 100% correct solutions resulting in a score of 1.0, consistent with the Pass@1 values ranging from 33% to approximately 44% improvement.

Mild off-policy matches track on-policy since the off-policy correction is a no-op; IS weights $\rho \approx 1$ when the action availability $\pi_b \approx \pi_\theta$, and stale $n \in \{1, 4, 16\}$ with diversified $T = 1.2$ continue to keep the two behaviors very close enough that the performance can be tracked on-policy.

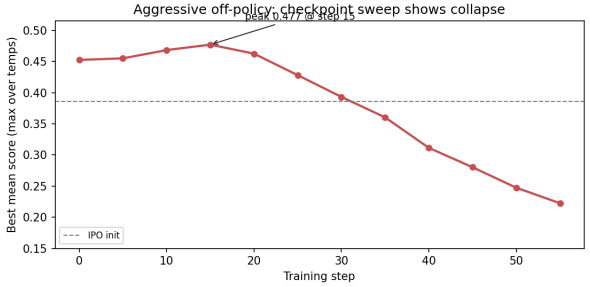
Aggressive Collapse Failure Mode: Qualitatively, the KL divergence between late-trained aggressive action pairs (e.g., $\pi_\theta \to \pi_b$) rises while the importance weights are erratic (e.g., impre-

Table 2: Optional hyperparameter sweeps (best checkpoint per setting).

Setting	**Mean**	**Pass@1**
Stale lag n		
n=1	0.475	45.1%
n=4	0.471	44.4%
n=16	0.472	44.6%
Replay ratio R		
R=1	0.467	44.2%
R=2	0.457	44.0%
R=4	0.490	46.1%
R=8	0.468	43.6%
$\rho_{\max}$ *(replay R=2)*		
$\rho_{\max}$=1	0.473	44.4%
$\rho_{\max}$=2	0.483	45.6%
$\rho_{\max}$=10	0.457	44.0%
$\rho_{\max}$=∞	0.474	44.4%



(a) Main ablations: mean vs. rollouts.



(b) Aggressive run: per-step best mean.

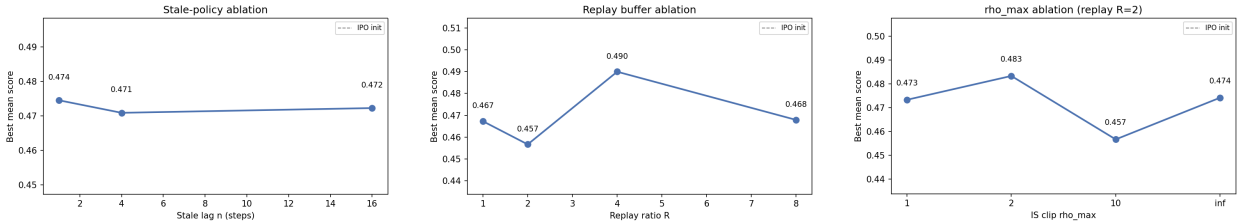
Figure 1: Quantitative summaries. Left: equal rollout budgets yield similar scores. Right: mid-training peak then collapse.

cision in calculating importance weights relative to KL divergence), and there is degradation of the formatting (i.e., missing tags, invalid equations). The checkpoint curve (Figure **??**) shows that the maximum performance occurs at the 15th timestep, with monotonic decay—consistent with accumulated off-policy bias under tight clipping with respect to token-level granularity and 4× replay updates.

Ablations as qualitative hypotheses. Replay $R = 8$ does worse than $R = 4$: excessive re-use of data without any new data would cause overfitting to stale advantages. $\rho_{\max} = 10$ with $R = 2$ does worse than $\rho = 2$: overly loose clipping may create catastrophic advantages.

6 Discussion

Limitations. It should be noted that all experiments were done on just one Model size (0.5B), only one task (Countdown), only one seed per config and only a 60 step budget which may not be representative of larger models/longer RL horizons; KL/grad-norm figures available through Modal



(a) Stale n (b) Replay R (c) $\rho_{\max}$

Figure 2: Optional sweep curves (best mean per hyperparameter value).

are unavailable because they were logged using W&B offline mode in W&B whereas in this work we did not utilize any form of prioritized replay access.

Broader Impacts. The reduction of compute from lower-cost RL fine-tuning allows for increased resources in alignment research but off-policy approaches that seem to work well partway into training could implement unstable policies without validation—and our case of specified collapsed is an example of that risk.

Encountered Problems. We did an aggressive run first (replay $R = 4$, token IS) but had not expected it would collapse, therefore utilizing moderate combined (replay $R = 2$, sequence IS), as well as documenting checkpoint sensitivity, resolved this issue. A `math.comb` glitch was present in our summary scripts that had negative results regarding relay checkpoint reloading, both would need to have debugged for any reliability in unified metrics.

7 Conclusion

RLOO within Qwen2.5-0.5B Countdown, with the following takeaways: You can use rollouts, but it is not a free lunch; from the perspective of mild settings, you will see that rollouts are equivalent to the on-policy version with the same rollout counts. However, from an aggressive-scenario point of view, this may render the smaller LM (language model) unstable; Please list (rollout count, gradient step count, and checkpoint sweep count) for ALL runs.

This includes future directions, which are: larger models (rollout costs are more significant), Adaptive ρ-clipping, Prioritized replay, Multi-seed runs, and plotting your learning-curve vs How many rollouts you have completed vs how many steps you have completed.

8 Team Contributions

- **Henok Tewolde:** Core implementation, Modal training/eval pipelines, metric aggregation, poster and final report drafting, debugging (replay reload, summary scripts).
- **Kennaissa Nabi:** Extension proposal, ablation design, experimental planning, poster content and presentation script, report writing and editing, milestone RLOO/IPO training support.

Changes from proposal. We added an explicit **mild combined** configuration after the first aggressive run collapsed; unified **extension_finalize** eval sweeps were not in the original plan but became necessary for fair comparison. Replay ablation used R=2 for mild combined (proposal

suggested $R \in \{1, 4, 8\}$; we ran the full grid in optional sweeps). Token-level IS was evaluated only in aggressive `combined`, not across all modes, due to instability.