Extended Abstract

Diversity-Aware RLOO for Pass@k Reasoning in Countdown

Jane Yang, Will Nathaniel Hansen

Motivation and problem statement. Many practical reasoning systems do not rely on a single language-model sample. They draw several candidate solutions, verify them with an external checker, and succeed if at least one candidate is correct. This setting is naturally evaluated by pass@k, but standard reinforcement learning fine-tuning usually assigns credit independently to each completion. In our Countdown arithmetic experiments, exact-reward RLOO improves single-sample correctness, yet it often spends the multi-sample budget on repeated versions of the same correct equation. The central question of this project is whether an RLOO objective can be shaped to preserve exact correctness while encouraging coverage of distinct correct solution modes.

Method and novelty. We introduce a diversity-aware reward for online RLOO. For each prompt, the policy samples a group of completions. A verifier first assigns the usual Countdown score: a completion receives full credit only if its final `<answer>` expression uses exactly the provided numbers and evaluates to the target. For each exact-correct completion, we extract the final answer expression and canonicalize it into an arithmetic signature by parsing its abstract syntax tree and normalizing commutative addition and multiplication. Correct completions with the same signature split one diversity bonus, while correct completions with distinct signatures receive separate bonuses:

$$r_i = \mathbf{1}[\text{correct}_i] + \lambda\, \mathbf{1}[\text{correct}_i]/\text{count}(\text{signature}_i).$$

The novelty is not a new verifier or inference-time reranker, but a training-time credit assignment rule that aligns RLOO more closely with the group-level goal used by pass@k evaluation.

Implementation and headline results. We implemented the method in the online RLOO training loop for Qwen2.5-0.5B on the Countdown task. Training samples groups of 8 responses per prompt, computes shaped rewards, applies leave-one-out advantages, and updates the policy with per-token importance weights, entropy regularization, and optional KL regularization. Evaluation uses 50 held-out Countdown prompts, 16 samples per prompt, temperature 1.0, top-$p = 1.0$, top-$k = -1$, and stop-after-answer decoding. Vanilla exact RLOO achieved the best pass@1, 0.550, but only 0.720 pass@16 and 1.54 unique correct signatures per problem. Our best diversity-aware run with $\lambda = 0.5$ achieved 0.780 pass@16 and 1.98 unique correct signatures, improving multi-sample coverage over vanilla RLOO and SFT while trailing IPO's 0.800 pass@16. A stronger $\lambda = 1$ setting raised diversity less usefully and reduced correctness, showing that the diversity reward must be tuned rather than maximized blindly.

Discussion, limitations, and conclusion. The results support the hypothesis that repeated correct rollouts are a real bottleneck for multi-sample reasoning, and that signature-aware rewards can recover some of the lost coverage. They also show a limitation: diversity pressure can reduce the mean number of correct samples, and our canonicalization only approximates mathematical equivalence. We ran a small held-out evaluation and report prompt-level standard errors, but did not complete multiple-seed training. The take-home message is that pass@k fine-tuning should reason about groups of samples, not only individual completions. Diversity-aware credit assignment is a simple, reproducible step in that direction, but future work should combine it with stronger algebraic equivalence checking, adaptive λ schedules, and larger multi-seed evaluations.

Diversity-Aware RLOO for Pass@k Reasoning in Countdown

Jane Yang
Department of Computer Science
Stanford University
yjane@stanford.edu

Will Nathaniel Hansen
Department of Computer Science
Stanford University
willnh@stanford.edu

CS224R Final Report

Abstract

Verifier-guided language-model reasoning is often evaluated by pass@k, the probability that at least one of k sampled solutions is correct. Standard RLOO fine-tuning optimizes per-completion reward and can therefore improve pass@1 while collapsing repeated samples onto the same correct expression. We study this mismatch on Countdown arithmetic and introduce Diversity-Aware RLOO, a reward-shaping extension that gives exact-correct completions a bonus for belonging to distinct canonical answer-expression signatures. The method is implemented inside the online RLOO loop: groups of sampled completions are verified, correct final expressions are canonicalized with an arithmetic AST signature, duplicate signatures split diversity credit, and the shaped rewards are used for leave-one-out policy-gradient updates. On 50 held-out Countdown prompts with 16 samples per prompt, the best diversity-aware run improves pass@16 from 0.720 for vanilla RLOO to 0.780 and raises mean unique correct signatures from 1.54 to 1.98, while preserving stronger pass@1 than SFT and IPO. The results show that group-aware reward design can recover multi-sample coverage, although excessive diversity pressure reduces correctness and the small evaluation leaves substantial uncertainty.

1 Introduction

Large language models are increasingly used as generators inside systems that can verify candidate answers. This pattern appears in mathematical reasoning, code generation, theorem proving, and program synthesis: the model proposes several candidates, and an automatic checker selects any candidate that satisfies the constraints. In these settings, the natural success metric is not only pass@1 but pass@k, the probability that at least one of k sampled completions is correct [4]. The evaluation rewards coverage across samples.

Standard RL fine-tuning, however, usually optimizes individual completions. PPO-style RLHF and RLOO both assign scalar rewards to sampled completions and update the model from those rewards [1, 2, 3]. This is effective when the deployment policy emits one answer, but it can be misaligned with multi-sample inference. If the model learns one high-probability correct expression and repeats it, expected single-sample reward can improve while the additional samples used for pass@k provide little new information.

Countdown arithmetic is a clean testbed for this issue. Each example provides a target and a small set of numbers. The model must produce an arithmetic expression using each number exactly once, and a deterministic verifier checks whether the expression evaluates to the target. This makes correctness cheap to measure, while multiple distinct expressions can solve the same prompt. In our milestone experiments, vanilla exact-reward RLOO substantially improved pass@1 but showed

weaker pass@16 than SFT and IPO. This suggested a mode-collapse failure: RL increased correctness for sampled completions, but not necessarily coverage over distinct ways to solve a problem.

This report asks three research questions. First, does vanilla RLOO collapse correct samples onto repeated answer signatures in Countdown? Second, can reward shaping based on distinct correct signatures improve pass@k coverage? Third, how large is the tradeoff between diversity and exact correctness? We hypothesize that rewarding distinct correct expression signatures will improve pass@16 and reduce duplicate correct samples, but that overly strong diversity pressure will reduce the number of correct samples.

2 Related Work

RLHF and policy-gradient fine-tuning are common post-training tools for language models. InstructGPT popularized PPO-based RLHF for aligning model outputs with human preferences [2], building on clipped policy optimization [1]. RLOO and related REINFORCE-style methods revisit simpler policy-gradient estimators for language-model alignment, using leave-one-out baselines across samples from the same prompt to reduce variance [3]. Our implementation follows this family: it samples a group of completions, computes scalar rewards, forms leave-one-out advantages, and performs a policy-gradient update.

For reasoning tasks, verifiers make RL particularly appealing. Work on GSM8K and verifier-guided reasoning showed that learned or rule-based checkers can score candidate reasoning traces and support both training and inference-time selection [6]. The Countdown task gives an even more direct verifier: an answer is correct if the final expression uses exactly the provided numbers and evaluates to the target. This allows us to focus on the reward-allocation problem rather than reward-model noise.

pass@k became a standard evaluation protocol through code-generation benchmarks such as Codex [4]. It measures the probability that at least one sampled program passes tests, and thus rewards useful diversity across samples. Self-consistency similarly improves reasoning by sampling multiple chains of thought and aggregating across them [5]. These methods show that sampling diverse candidate solutions can be more powerful than relying on a single greedy output. They are primarily inference-time ideas; our project instead changes the training reward so that the policy itself receives credit for covering multiple correct modes.

The closest conceptual connection is to diversity-promoting objectives in sequence generation and exploration bonuses in RL. However, generic diversity rewards can encourage incorrect or merely different outputs. Our method is narrower: it only rewards diversity among completions that already pass the exact verifier. This keeps the novelty grounded in the course project's RLOO pipeline and the pass@k mismatch observed in Countdown.

3 Method

3.1 Base RLOO Training Loop

We fine-tune Qwen2.5-0.5B with online RLOO. For each prompt x, the current policy samples a group of $G = 8$ completions $\{y_i\}_{i=1}^{G}$. Each completion is scored by the Countdown verifier, producing raw rewards s_i. The training code flattens prompt-major groups into tokenized examples, computes sequence log-probabilities under the current policy, and forms leave-one-out advantages

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

where r_i is either the exact reward or our shaped reward. The scalar objective is

$$\mathcal{L} = -\frac{1}{G}\sum_{i=1}^{G} w_i A_i \log \pi_\theta(y_i \mid x) - \alpha H(\pi_\theta) + \beta \,\mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}),$$

where w_i is a detached importance weight from sampler log-probabilities, H is mean response-token entropy, and the KL term is optional. In our main RLOO runs we use per-token importance ratios clipped in log-space, entropy coefficient 0.001, KL coefficient 0.001, AdamW, constant learning rate, and checkpoint handoff between a vLLM sampling worker and a PyTorch update worker.

3.2 Verifier and Answer Signatures

The Countdown verifier extracts the final `<answer>...</answer>` span. A completion receives reward 1.0 only if the expression uses exactly the provided numbers with the correct multiplicity and evaluates to the target. If the answer tags are present but the equation is invalid or wrong, it receives a format score of 0.1. If no answer tag is found, it receives 0.0.

For diversity-aware training and analysis, we define a canonical signature for exact-correct completions. We extract the final answer expression, remove whitespace, parse it as a Python arithmetic expression, and recursively canonicalize the abstract syntax tree. Numeric constants are normalized, and addition and multiplication are flattened and sorted so that commutative variants such as `44 + 19 + 35` and `35 + 44 + 19` share one signature. Non-commutative operations preserve order, so `98 - (45 - 20)` and `(98 + 20) - 45` can remain distinct signatures even if they are algebraically equivalent.

This signature is an approximation to solution mode. It is intentionally simple and reproducible: it detects common duplicate surface forms without requiring a full symbolic algebra system. The limitation is that it can overcount or undercount mathematical equivalence, which we discuss later.

3.3 Diversity-Aware Reward

Vanilla exact RLOO uses $r_i = \mathbf{1}[\mathrm{correct}_i]$. We instead assign correct completions a diversity bonus based on how many samples in the same prompt group share their canonical signature:

$$r_i = \mathbf{1}[\mathrm{correct}_i] + \lambda \frac{\mathbf{1}[\mathrm{correct}_i]}{\mathrm{count}(\mathrm{signature}_i)}.$$

Incorrect completions receive no diversity bonus. If a group contains one correct solution mode repeated eight times, every correct sample receives only $\lambda/8$ extra credit. If the group contains two distinct correct signatures with four samples each, each correct sample receives $\lambda/4$, and the total group bonus is larger because the group covered more correct modes. This makes distinct correct expressions more valuable without rewarding unverified diversity.

Pseudocode. For each prompt group, the reward computation is:

Input: verifier scores $s_{1:G}$, responses $y_{1:G}$, diversity weight λ.
1. Set $c_i \leftarrow \mathbf{1}[s_i \geq 1.0]$.
2. For every i with $c_i = 1$, extract the final `<answer>` expression and compute canonical signature σ_i.
3. Count n_σ, the number of correct samples in the group with each signature σ.
4. Return $r_i \leftarrow c_i + \lambda c_i / n_{\sigma_i}$ for correct samples, and $r_i \leftarrow 0$ otherwise.
5. Use $r_{1:G}$ in the standard leave-one-out RLOO update.

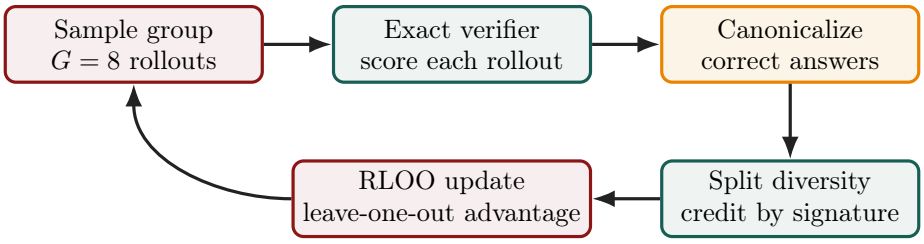

Figure 1: Training-time reward pipeline. Diversity is only rewarded after exact verification, and duplicate correct answer signatures split credit.

Table 1: Countdown evaluation with 16 samples per prompt. Parentheses report standard error over 50 prompt-level pass@k values.

Method	pass@1	pass@4	pass@8	pass@16	Correct	Unique sig.	Duplicate	Zero-correct
SFT	0.281±0.031	0.614±0.055	0.720±0.060	0.760±0.061	4.50	1.40	3.10	12
IPO	0.270±0.033	0.584±0.052	0.720±0.055	0.800±0.057	4.32	1.72	2.60	10
Vanilla RLOO	0.550±0.059	0.681±0.063	0.709±0.064	0.720±0.064	8.80	1.54	7.26	14
Diverse RLOO, $\lambda = 1$	0.328±0.042	0.576±0.063	0.635±0.065	0.680±0.067	5.24	1.76	3.48	16
Diverse RLOO, $\lambda = 0.5$	0.469±0.050	0.687±0.059	0.743±0.058	0.780±0.059	7.50	1.98	5.52	11

4 Experimental Setup

Task and data. We evaluate on Countdown arithmetic with 3–4 input numbers. Each prompt asks the model to produce a reasoning trace in `<think>` tags and a final expression in `<answer>` tags. The expression may use $+$, $-$, $*$, and $/$, and each provided number must be used exactly once. We use the `asingh15/countdown_tasks_3to4` test split for evaluation and report results on 50 held-out prompts.

Models and baselines. The base model is Qwen2.5-0.5B. We compare five checkpoints: supervised fine-tuning (SFT), IPO preference fine-tuning, vanilla exact-reward RLOO, diversity-aware RLOO with $\lambda = 1$, and a rescue diversity-aware RLOO run with $\lambda = 0.5$. The SFT run used learning rate $5 \cdot 10^{-5}$ for 6 epochs. The IPO checkpoint used $\beta = 0.1$ and the evaluated run name indicates learning rate 10^{-5} for 3 epochs. The RLOO runs start from the SFT policy, use group size 8, batch size 128, gradient accumulation 128, constant learning rate 10^{-5}, entropy coefficient 0.001, KL coefficient 0.001, and per-token importance weighting.

Evaluation protocol. For each checkpoint, we sample 16 responses per held-out prompt with temperature 1.0, top-$p = 1.0$, top-$k = -1$, maximum 1024 tokens, and stop-after-answer decoding. We report pass@1, pass@4, pass@8, pass@16, mean number of exact-correct samples per prompt, mean unique correct signatures per prompt, mean duplicate correct samples per prompt, and number of prompts with zero correct samples. Standard errors are computed over the 50 prompt-level pass@k values. We did not complete multiple training seeds, so these standard errors quantify evaluation-set variability rather than full training-run variance.

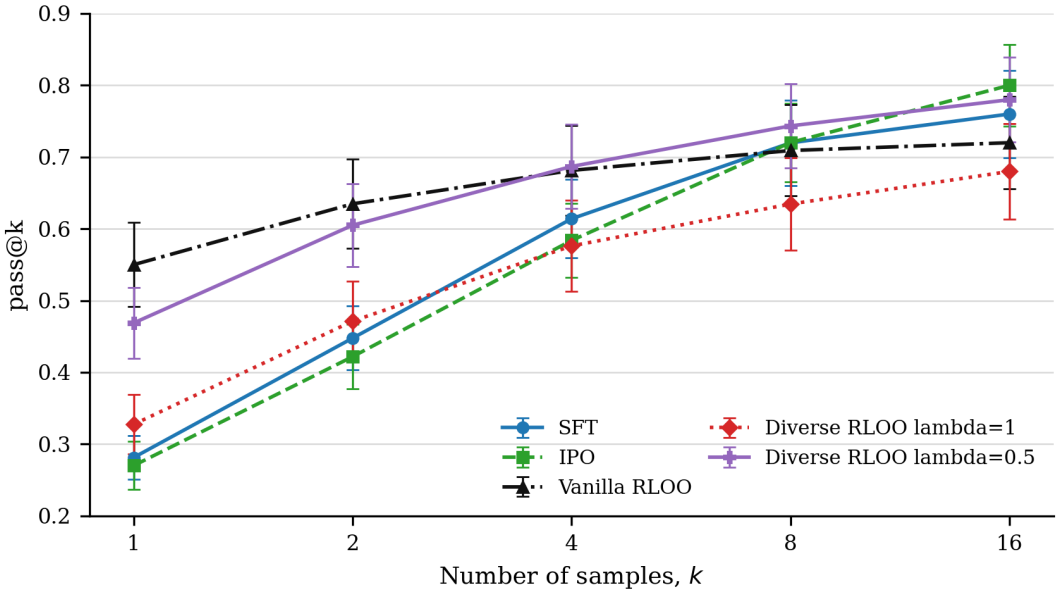

Figure 2: Pass@k curves with prompt-level standard errors. Diversity-aware RLOO with $\lambda = 0.5$ recovers pass@8 and pass@16 relative to vanilla RLOO while preserving much of the pass@1 gain.

5 Results

5.1 Quantitative Evaluation

Table 1 shows the main tradeoff. Vanilla RLOO is the strongest pass@1 model: it raises pass@1 to 0.550, nearly doubling the SFT and IPO baselines. However, it has worse pass@16 than both SFT and IPO. Its mean correct count is high, 8.80 out of 16 samples, but its unique correct signature count is only 1.54. This is the collapse pattern that motivated the extension: many samples are correct, but many are duplicates.

The best diversity-aware run is $\lambda = 0.5$. It improves pass@16 from 0.720 for vanilla RLOO to 0.780, improves pass@8 from 0.709 to 0.743, and raises unique correct signatures from 1.54 to 1.98. It also reduces zero-correct prompts from 14 to 11. This supports the hypothesis that signature-aware reward shaping can improve multi-sample coverage.

The $\lambda = 1$ run is a useful negative result. It reduces duplicate correct samples from 7.26 to 3.48, but it also reduces mean correct samples from 8.80 to 5.24 and drops pass@16 to 0.680. The method therefore does not work by maximizing diversity pressure. It works only when the diversity bonus is small enough that exact correctness remains the dominant signal.

Figures 2 and 3 visualize the same result. The vanilla RLOO curve starts high at pass@1 but saturates early because additional samples are often redundant. The $\lambda = 0.5$ run gives up some single-sample accuracy but continues to gain with larger k, which is the desired behavior for multi-sample verifier-guided deployment.

5.2 Qualitative Analysis

The qualitative examples clarify why the aggregate numbers move this way. Table 2 lists representative held-out prompts and the number of exact-correct and unique-signature completions among 16 samples.

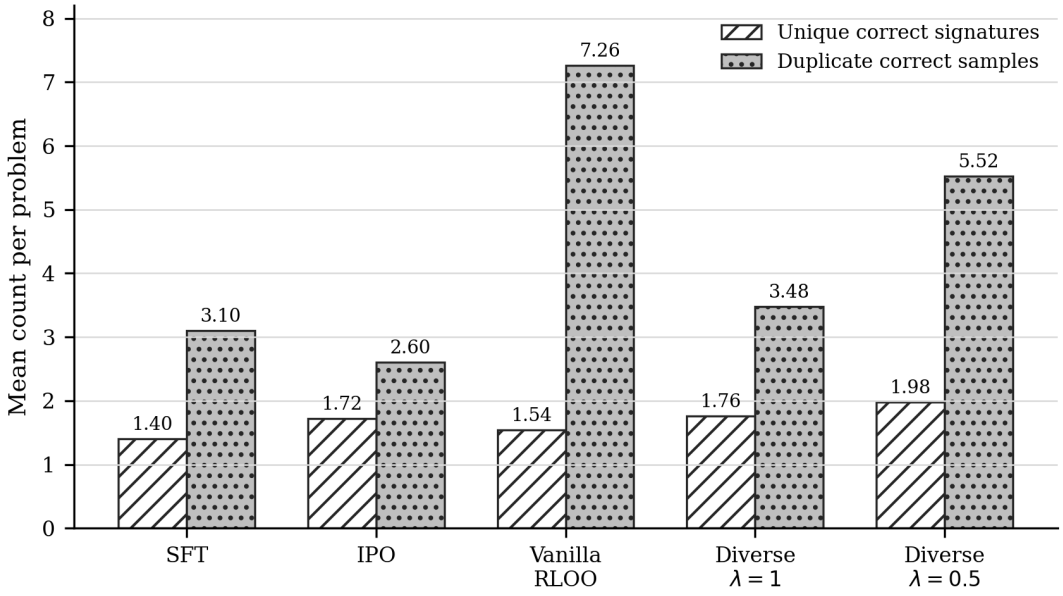

Figure 3: Mean unique correct signatures and duplicate correct samples across 16 generations. Vanilla RLOO produces many correct samples but a large duplicate count.

These cases show two mechanisms. First, diversity-aware credit can rescue prompts where vanilla RLOO has no correct samples. Second, on easier prompts, vanilla RLOO can already be correct almost all the time, but it still repeats one expression family. In that regime, diversity reward converts some duplicated samples into alternative correct forms. The cost is visible in the third row: lower duplicate count sometimes comes with fewer exact-correct samples.

The examples also expose a measurement limitation. The signature system normalizes commutative addition and multiplication, but it is not a full algebraic simplifier. For example, `98 - (45 - 20)` and `(98 + 20) - 45` are mathematically equivalent but can be counted as distinct signatures because subtraction order remains structurally different. Thus the signature metric should be read as a practical proxy for expression diversity, not a perfect equivalence class.

6 Discussion

The main difficulty was maintaining correctness while adding diversity pressure. A naive group-level coverage bonus can reward a group for having any correct sample, but it does not distinguish one correct mode from many correct modes. The signature-aware reward is more targeted, yet the $\lambda = 1$ result shows that even targeted diversity pressure can hurt. The best setting, $\lambda = 0.5$, works because it makes diversity a secondary signal.

There are several limitations. The evaluation uses only 50 held-out prompts and one training seed per configuration. We report standard errors over prompts, but the true run-to-run variance is unknown. The canonical signature function is also incomplete: it catches common duplicate surface forms but does not prove algebraic equivalence. Finally, the model sometimes generates extra text or repeated answer tags after finding a valid answer; stop-after-answer decoding helps evaluation, but training-time rollouts can still contain noisy tails.

The broader impact is mostly about reliability of verifier-guided reasoning systems. Better

Table 2: Representative qualitative cases. Counts are exact-correct samples / unique correct signatures.

Prompt	Counts	Interpretation
Target 89; nums [94, 89, 94]	Vanilla 12/2; diverse 11/5	The rescue run keeps similar correctness but finds more forms, including `89 - (94 - 94)`, `(89 - 94) + 94`, and `(94 / 94) * 89`.
Target 98; nums [67, 21, 31, 20]	Vanilla 0/0; diverse 6/2	Vanilla RLOO misses the problem entirely, while the diversity-aware run finds both `(21 - 20) * 67 + 31` and `(21 - 20) * 31 + 67`.
Target 73; nums [45, 20, 98]	Vanilla 16/1; diverse 11/4	Vanilla solves every sample but repeats essentially one subtraction pattern. The diverse run sacrifices five correct samples but covers `(20 - 45) + 98`, `98 - (45 - 20)`, and `(98 + 20) - 45`.
Target 57; nums [4, 50, 92, 11]	Vanilla 13/1; diverse 12/3	The diversity-aware run keeps nearly the same correctness and adds signatures such as `(92 + 4) - 50 + 11` and `(92 + 11) - 50 + 4`.

multi-sample coverage can make small models more useful under compute budgets where sampling 8–16 candidates is affordable. At the same time, optimizing against an automatic verifier can encourage formatting artifacts or reward hacking if the verifier is incomplete. For tasks beyond Countdown, diversity rewards should be paired with strong correctness checks and monitoring for degenerate output patterns.

7 Conclusion

This project shows that vanilla exact-reward RLOO can improve single-sample Countdown accuracy while weakening multi-sample coverage through duplicate correct solutions. A simple diversity-aware reward that splits bonus credit among duplicate answer signatures improves pass@16 and unique correct signatures when tuned conservatively. The best run, $\lambda = 0.5$, raises pass@16 from 0.720 to 0.780 relative to vanilla RLOO and reduces zero-correct prompts from 14 to 11, although it still trails IPO on pass@16 and trails vanilla RLOO on pass@1.

The take-home message is that pass@k systems need training objectives that reason about groups of samples. Future work should run multiple seeds, evaluate on more prompts, replace AST signatures with stronger symbolic equivalence checks, and use adaptive schedules that start with exact correctness before gradually increasing diversity pressure.

8 Team Contributions

Jane Yang. Implemented and maintained the Modal training/evaluation workflow, managed W&B logging and remote checkpoint/evaluation runs, ran the SFT-to-IPO/RLOO experiments, implemented and evaluated the diversity-aware reward shaping pipeline, computed the pass@k and signature-diversity summaries, generated the report figures, and prepared the final report materials.

Will Nathaniel Hansen. Helped implement and debug the IPO/RLOO baselines, contributed

to the RLOO training design and stability checks, reviewed pass@k outputs and qualitative rollouts, contributed to extension-result analysis, and wrote the final report.

Changes from proposal. The extension was narrowed from a broad pass@k group-success reward to a signature-level diversity reward after the milestone results showed that vanilla RLOO's problem was not merely missing group-level success, but duplicating the same correct expression. We also added the $\lambda = 0.5$ rescue run after $\lambda = 1$ reduced correctness too much. These changes preserve the proposal's goal of aligning RLOO with pass@k while making the credit assignment more diagnostic and reproducible.

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

[2] L. Ouyang et al. Training language models to follow instructions with human feedback. arXiv:2203.02155, 2022.

[3] A. Ahmadian et al. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. arXiv:2402.14740, 2024.

[4] M. Chen et al. Evaluating large language models trained on code. arXiv:2107.03374, 2021.

[5] X. Wang et al. Self-consistency improves chain of thought reasoning in language models. arXiv:2203.11171, 2022.

[6] 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:2110.14168, 2021.

A Additional Ablation Notes

Earlier diverse-pass@k trials explored stronger diversity pressure and intermediate checkpoints. A $\lambda = 2$ fast run had 2.18 unique correct signatures but only 0.700 pass@16, while a step-50 $\lambda = 1$ run reached 0.740 pass@16 before later degradation. These runs motivated the final $\lambda = 0.5$ rescue experiment.

B Reproducibility Artifacts

The reward implementation is in `rloo_trainer/reward_shaping.py`. The RLOO orchestration and update worker are in `rloo_trainer/rloo.py` and `rloo_trainer/rloo_update_worker.py`. The evaluation summarizer is `extension_project/analyze_passk_diversity.py`. The report figures and standard-error CSV are generated by `extension_project/make_report_figures.py`.