# Extended Abstract

## Making RLOO Learn From Better Signals

Curriculum Scheduling and Verification Commit Contrast for Countdown Reasoning

Zayn Malhotra Ziyi Ding

**Motivation and problem.** Reinforcement learning with verifiable rewards is attractive for mathematical reasoning because correctness can be checked automatically. Recent systems such as DeepSeek-R1 show that verifier-based reinforcement learning can elicit reasoning behavior at scale, but the default CS224R Countdown setting is deliberately much smaller and noisier: we fine-tune a Qwen2.5-0.5B policy to solve arithmetic puzzles where the verifier only gives full reward when the final expression uses each number exactly once and equals the target. This creates a sparse credit assignment problem. In our SFT generations, many rollouts are parseable and some even contain a correct answer, but the standard verifier only evaluates the final `<answer>` span. A rollout that discovers a correct expression early and then drifts into a wrong final answer is treated almost the same as a rollout that never knew the solution. Our project asks whether we can make RLOO more sample efficient by improving the signal available to the policy without changing the task, adding a learned reward model, or using extra labels.

**Method and novelty.** We developed two extensions that address different failure modes of sparse-reward RLOO. Curriculum RLOO changes the prompt distribution rather than the reward. For each Countdown prompt, we compute an arithmetic solution count by exhaustively enumerating valid expressions with exact rational arithmetic. Prompts with many valid derivations are treated as easier. Training then proceeds easy to hard: the sampler first draws from the easiest 30 percent of prompts, expands to the easiest 70 percent, and finally uses the full training distribution. This is a task-specific curriculum that keeps early RLOO groups near the model's competence frontier, where sampled rollouts are more likely to contain reward variation. VCC-RLOO changes how rollouts are used. We sample full responses without stopping at `</answer>`, scan every answer tag with the deterministic Countdown verifier, and identify the first answer that solves the prompt. If a response later rambles or abandons that answer, we create a verifier-generated preference pair: the chosen response is the truncated "commit" ending at the first correct answer, while the rejected response is the full continuation or a same-prompt rollout that never verifies. We optimize the usual leave-one-out policy-gradient loss together with a small DPO-style contrastive term. The novelty is that both extensions are model-free and verifier-generated. Curriculum RLOO uses a difficulty signal from the search space of the task itself, and VCC-RLOO turns partial correctness inside a rollout into preference data for online RL.

**Implementation and headline results.** We trained from the staff SFT checkpoint on the staff Countdown dataset using Modal H100 workers and vLLM sampling. Evaluation used the unchanged staff Countdown verifier on 50 held-out test prompts, 16 sampled responses per prompt, temperature 0.6, top-p 0.95, top-k 20, and max length 1024. The SFT baseline achieved 0.3038 rollout accuracy and 0.7400 oracle pass@16. IPO improved rollout accuracy to 0.3750 and pass@16 to 0.7600. The RLOO baseline reached 0.4963 rollout accuracy, 0.6756 average pass@k, and 0.7200 pass@16. VCC-RLOO with outcome reward, averaged log-probability contrast, coefficient $c = 0.01$, and checkpoint selection at step 60 reached 0.3262 rollout accuracy and the strongest pass@16 in the main-method comparison, 0.7800. Among the curriculum variants, the selected Hybrid Curriculum checkpoint produced the strongest single-sample result, reaching 0.5150 pass@1, although its pass@16 was lower at 0.7000. These results show a clean tradeoff: curriculum scheduling improves per-rollout quality, while VCC-RLOO preserves coverage and exploits recoverable answer-abandonment cases.

**Discussion and conclusion.** The main lesson is that sparse verifier rewards are not always as sparse as they first appear. The rollout structure often contains extra information about the policy's capabilities. Curriculum RLOO extracts signal before generation by choosing prompts that are more likely to generate useful contrast. VCC-RLOO extracts signal after generation by identifying correct commits that final-answer scoring discards. There are important limitations. The test set is small, most conditions were not run with enough seeds for strong statistical claims, and solution count is only a proxy for difficulty. VCC-RLOO also depends on the model finding at least one correct answer inside the rollout, so it cannot help on prompts where all sampled traces fail. Still, the two methods are complementary in a useful way. A natural next step is to train VCC-RLOO inside the curriculum schedule, so the curriculum increases the rate of recoverable answers and VCC-RLOO teaches the model to commit to them. Overall, our final extension moved away from the proposal's less novel dense reward shaping plan toward two verifier-grounded mechanisms that better match the observed failure modes of small-model RLVR.

# Making RLOO Learn From Better Signals: Curriculum Scheduling and Verification Commit Contrast for Countdown Reasoning

Zayn Malhotra Ziyi Ding
Stanford University, CS224R

**Abstract**

We study reinforcement learning with verifiable rewards on the Countdown arithmetic task using a Qwen2.5-0.5B policy. Standard RLOO receives sparse reward from a final-answer verifier, which wastes signal when prompts are too hard early in training or when rollouts find a correct answer but later abandon it. We introduce two verifier-grounded extensions. Curriculum RLOO orders prompts by an exhaustive arithmetic solution-count difficulty proxy and trains easy to hard. VCC-RLOO scans full rollouts for the first verified answer and adds a small contrastive loss that prefers committing to that answer over continuing into a worse response. On the held-out Countdown test set, the RLOO baseline reaches 0.4963 rollout accuracy versus 0.3038 for SFT, and Selected Hybrid Curriculum further improves pass@1 to 0.5150. In the main-method comparison, VCC-RLOO gives the strongest coverage result, reaching 0.7800 oracle pass@16 versus 0.7400 for SFT and 0.7600 for IPO. These results suggest that small-model RLVR can benefit from better use of verifier structure, especially when the method is matched to the observed failure mode.

## 1 Introduction

Verifier-based reinforcement learning is becoming a central recipe for reasoning models. When the final answer can be checked automatically, as in arithmetic, code, and many logic tasks, the reward no longer needs to come from a learned preference model. This is the promise behind reinforcement learning with verifiable rewards, or RLVR. It lets us train from large numbers of sampled attempts while keeping evaluation objective and cheap.

The promise is not the whole story. In the CS224R Countdown task, a model is given a target and a small set of numbers, and must output an expression that uses every number exactly once and evaluates to the target. The verifier is simple: extract the final `<answer>` span, check the numbers, safely evaluate the expression, and return full reward only for exact correctness. This reward is clean, but it is sparse. A small model may sample many wrong attempts for the same prompt, which gives RLOO little useful within-group contrast. The same model may also find a correct expression early, continue generating, and end with an incorrect final answer. Standard final-answer scoring discards that partial success.

Our original proposal focused on dense reward shaping: format, number usage, evaluability, proximity to the target, and exact correctness. After proposal feedback and additional experiments, we changed direction. The dense reward idea was plausible, but it was not novel enough and it did not directly address the most interesting behavior we observed in rollouts. The final project instead studies two extensions that use the verifier in more structured ways:

1. **Can a task-specific curriculum improve RLOO by choosing prompts that are likely to yield informative reward contrast?**
2. **Can verifier-detected answer commits recover useful signal from rollouts that standard final-answer scoring treats as wrong?**
3. **Do these extensions improve the same metric, or do they change different parts of the accuracy and coverage tradeoff?**

The answer is mixed but useful. The RLOO baseline is already strong, and Selected Hybrid Curriculum gives the strongest per-rollout policy. In the main-method comparison, VCC-RLOO gives the strongest oracle pass@16 and a direct qualitative explanation for where the gain comes from. Together they show that small-model RLVR is not only about changing the optimizer. The prompt distribution and the internal structure of a rollout both matter.

## 2 Related Work

**RLHF, RLOO, and direct preference methods.** PPO has long been the standard policy optimization method in RLHF [Schulman et al., 2017], but recent work argues that simpler REINFORCE-style methods can be competitive for language-model alignment. Ahmadian et al. [2024] revisit REINFORCE-style optimization and show that many PPO components are not always necessary in the RLHF setting. This motivates our choice of RLOO, which uses multiple responses per prompt and a leave-one-out baseline to reduce variance while avoiding a learned value function. Direct preference methods such as DPO [Rafailov et al., 2023] and IPO [Azar et al., 2023] optimize pairwise preferences without online RL. Our VCC-RLOO auxiliary loss borrows the preference-optimization form, but the pairs are not human labeled. They are generated online by the Countdown verifier from the model's own rollouts.

**Verifiable rewards and mathematical reasoning.** Work on mathematical verifiers shows that checking is often easier than generation. Cobbe et al. [2021] use verifiers to rank candidate solutions on GSM8K. Lightman et al. [2024] compare outcome supervision with process supervision and show the value of step-level feedback, though their strongest process reward model relies on human-labeled intermediate steps. DeepSeek-R1 demonstrates that large-scale RL with rule-based rewards can elicit strong reasoning behavior [Guo et al., 2025]. Our setting is smaller and stricter. We use a 0.5B model, a small warm-start dataset, and an arithmetic verifier. The project asks which parts of the verifier can still be exploited when the model is not already strong enough for sparse outcome RL to work easily.

**Curriculum learning.** Curriculum learning organizes examples from easy to hard, following the intuition that optimization can improve when the learner first sees examples near its current ability [Bengio et al., 2009]. Self-paced learning extends this idea by letting the training process select easier examples first [Kumar et al., 2010]. Our curriculum is not learned from model confidence. It uses the combinatorics of Countdown itself: prompts with many valid arithmetic derivations are easier because random or weakly guided samples have more ways to land on a correct answer.

**Positioning of our novelty.** Prior RLVR work often changes the policy optimizer, scales the model, or trains an external reward model. Our extensions do not do that. Curriculum RLOO changes which prompts enter the RLOO groups as training progresses. VCC-RLOO changes how full rollouts are converted into rewards and preferences by detecting the first verified answer. The novelty is the combination of online RLOO with task-derived curriculum difficulty and verifier-generated commit preferences.

## 3 Method

### 3.1 Countdown and RLOO

Each prompt $x$ contains a target $t$ and a multiset of numbers $N$. A response $y$ receives full reward if the final answer span is an arithmetic expression that uses each number in $N$ exactly once and evaluates to $t$. Let $K$ responses be sampled for a prompt, with rewards $r_1, \ldots, r_K$. The RLOO baseline for response $i$ is

$$b_i = \frac{1}{K-1} \sum_{j \neq i} r_j,$$

and the policy-gradient objective is

$$\mathcal{L}_{RLOO} = -\frac{1}{K} \sum_{i=1}^{K} (r_i - b_i) \log \pi_\theta(y_i \mid x),$$

with entropy and KL regularization in the implementation. This objective only becomes informative when the group contains reward variation. If all $K$ responses are wrong, the leave-one-out advantages are nearly useless. If a response finds a correct answer and later abandons it, the final-answer reward can also erase useful evidence.

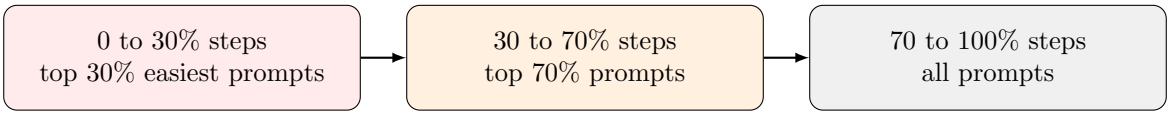


Figure 1: Curriculum RLOO changes the prompt distribution while keeping the RLOO objective fixed. The difficulty proxy is the number of valid arithmetic derivations for the prompt.

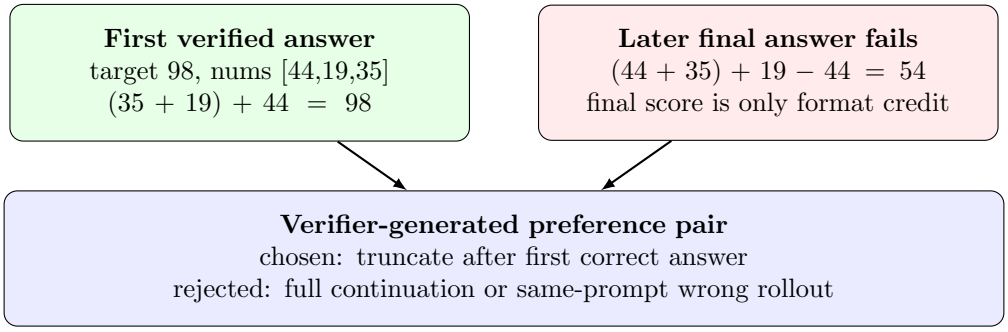


Figure 2: VCC-RLOO recovers signal from rollouts where final-answer scoring discards an earlier correct answer.

### 3.2 Curriculum RLOO

Curriculum RLOO keeps the RLOO loss fixed and changes the sampler. For each training prompt, we compute a solution-count difficulty proxy by recursively combining pairs of current values with $+, -, \times, \div$ until one value remains. We use exact rational arithmetic to avoid floating-point artifacts. The search returns the number of derivations that evaluate to the target, capped at 128 for efficiency. Prompts are sorted in descending solution count.

Training uses three stages. During the first 30 percent of updates, batches are sampled from the easiest 30 percent of prompts. During the next 40 percent, batches are sampled from the easiest 70 percent. During the last 30 percent, the full training set is used. The goal is not to make evaluation easier. Evaluation remains the full held-out test set. The goal is to increase the chance that early RLOO groups contain at least one correct rollout, which makes the leave-one-out baseline meaningful.

As a concrete example, for target 24 with numbers $[1, 2, 6, 8]$, the search finds 75 derivations, including $8 \times (6 - (1 + 2)) = 24$, so the prompt enters early. For target 24 with numbers $[1, 3, 4, 6]$, the search finds only one derivation, $6/(1 - 3/4) = 24$, so the prompt enters late.

### 3.3 Verification Commit Contrast

VCC-RLOO targets answer abandonment. We define an answer span as any text between `<answer>` and `</answer>`. Instead of stopping generation at the first answer tag, we sample full rollouts. For each rollout, we scan answer spans in order and use the deterministic Countdown verifier to find the first answer that is actually correct. If such an answer exists, the prefix ending at that answer is a clean commit response.

There are two ways to form a rejected response. If the same rollout continues after the first correct answer, the full continuation is rejected. If the prompt group also contains a rollout that never verifies, that wrong rollout can be rejected too. We cap the number of contrastive pairs per prompt to keep the auxiliary loss small.

The contrastive objective is DPO-style:

$$\mathcal{L}_{\mathrm{VCC}} = -\log \sigma \left( \beta \left[ \log \pi_\theta(y^+|x) - \log \pi_\theta(y^-|x) - \log \pi_{\mathrm{ref}}(y^+|x) + \log \pi_{\mathrm{ref}}(y^-|x) \right] \right),$$

where $y^+$ is the committed response and $y^-$ is the rejected continuation. In the strongest outcome+VCC-RLOO run, we used length-averaged log probabilities, $\beta = 0.1$, and coefficient $c = 0.01$. The full loss is

$$\mathcal{L} = \mathcal{L}_{RLOO} + c\mathcal{L}_{\mathrm{VCC}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}} - \lambda_H H(\pi_\theta).$$

We also tried a commit-aware scalar reward. If the first correct answer ends at character position $e$ in a response of length $L$, the reward is $1 - (L - e)/L$, lower bounded by the format score. This penalizes text after

the first correct answer. In practice, the pure commit-reward variant was not the best final method. The best VCC-RLOO result kept the standard outcome reward and used the contrastive commit loss as an auxiliary term.

# 4 Experimental Setup

**Task and data.** All experiments use the Countdown task from `asingh15/countdown_tasks_3to4`. The SFT warm start follows the staff default setup. RLOO-style methods train on the Countdown train split and evaluate on the held-out test split. The tight VCC-RLOO sweep used `train[200:]` for training and `train[:200]` for validation checkpoint selection, then evaluated the selected checkpoint on the test split.

**Models and baselines.** The base policy is Qwen2.5-0.5B. The baselines are SFT, IPO, the RLOO baseline, and RLOO controls without the proposed auxiliary signal. IPO is included because it is the pairwise preference baseline from the default project.

**Training details.** For RLOO and VCC-RLOO runs, we initialized both policy and reference from the four-epoch SFT checkpoint. We used AdamW, learning rate $10^{-6}$ for the VCC experiments, constant schedule, group size 4, batch size 8, gradient accumulation 4, entropy coefficient 0.001, KL coefficient 0.001, max generation length 1024, and H100 Modal workers. The first VCC sweep saved checkpoints every 20 steps. The tight sweep trained for 80 steps, saved every 10 steps, and swept coefficients $c \in \{0.005, 0.0075, 0.01, 0.0125, 0.015\}$ across two seeds with matching no-contrast baselines.

**Evaluation.** We use the unchanged Countdown verifier for all reported metrics. Each eval JSON contains 50 prompts and 16 sampled responses per prompt. Sampling used temperature 0.6, top-p 0.95, top-k 20, min-p 0, and max length 1024. Rollout accuracy is the fraction of all sampled responses with verifier score 1.0; equivalently, it is pass@1 under this evaluation. Oracle pass@16 is the fraction of prompts for which at least one of 16 samples is correct. Pass@k curves use the standard estimator $1 - \binom{n-c}{k}/\binom{n}{k}$, where $n = 16$ and $c$ is the number of correct samples for a prompt. Avg. pass@k averages this estimator over all $k = 1, \ldots, 16$. Mean reward is separate: it averages the raw verifier scores, including partial format credit.

# 5 Results

## 5.1 Main Quantitative Results

| Method | Avg. pass@k | pass@1 | pass@2 | pass@4 | pass@8 | pass@16 |
|---|---|---|---|---|---|---|
| Baseline RLOO | 0.6756 | 0.4963 | 0.5935 | 0.6542 | 0.6956 | 0.7200 |
| IPO | **0.6831** | 0.3750 | 0.5350 | 0.6524 | 0.7178 | 0.7600 |
| Curriculum RLOO step90 | 0.6580 | 0.5088 | 0.5993 | 0.6406 | 0.6699 | 0.7000 |
| SFT | 0.6454 | 0.3038 | 0.4567 | 0.5943 | 0.6883 | 0.7400 |
| VCC-RLOO | 0.6732 | 0.3263 | 0.4847 | 0.6174 | 0.7101 | **0.7800** |
| Adaptive Curriculum | 0.6298 | 0.4638 | 0.5447 | 0.5991 | 0.6472 | 0.6800 |
| Selected Hybrid Curriculum | 0.6703 | **0.5150** | **0.6077** | **0.6577** | 0.6890 | 0.7000 |

Table 1: Held-out test pass@k results. Each row uses 50 prompts and 16 sampled responses per prompt. Avg. pass@k averages pass@k over all $k = 1, \ldots, 16$; pass@1 is the single-rollout accuracy reported by the milestone grader. Baseline, IPO, SFT, and VCC-RLOO rows are regenerated from checked-in eval JSONLs. Curriculum-variant rows are summaries from Ziyi's shared eval zip, with source filenames recorded in `tables/curriculum_variant_summary.csv`.

Table 1 shows the core result. IPO has the highest average over the pass@k curve, reaching 0.6831. The RLOO baseline is already strong, reaching 0.4963 pass@1. Curriculum step90 improves pass@1 to 0.5088, and Selected Hybrid Curriculum gives the highest single-sample result overall at 0.5150. The cost is diversity or coverage: the curriculum variants have pass@16 between 0.6800 and 0.7000, below the RLOO baseline's 0.7200. VCC-RLOO shows the opposite pattern. Its pass@1 is low at 0.3263, but it reaches the highest pass@16 in the main

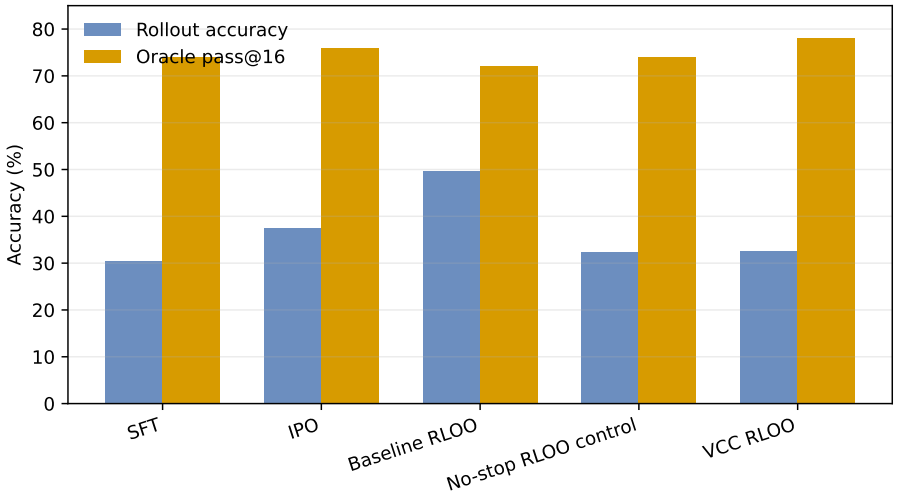


Figure 3: Single-rollout accuracy and oracle pass@16 show different strengths for the checked-in JSONL rows. Baseline RLOO improves the average sample. VCC-RLOO gives the strongest pass@16 coverage in the main-method comparison.

comparison, 0.7800. This suggests that VCC-RLOO did not turn the policy into the best single-shot solver, but it helped preserve or increase the chance that at least one sampled response solves the prompt.

Figure 4 makes the tradeoff clearer. Baseline RLOO and the curriculum variants have higher pass@1 estimates because many individual samples are correct. But their curves are flatter, meaning additional samples add less new coverage. VCC and IPO start lower but preserve more of the pass@k climb. This matters for Countdown because pass@k measures whether the model has retained multiple viable solution modes, not only whether the most common sample is correct.

## 5.2 VCC Ablations and Tight Sweep

| Method | Rollout acc. | Mean reward | Oracle pass@16 |
|---|---|---|---|
| SFT | 0.3038 | 0.3574 | 0.7400 |
| RLOO baseline | 0.3063 | 0.3611 | 0.7800 |
| Commit reward only | 0.2925 | 0.3481 | **0.8000** |
| Commit reward + VCC | 0.3125 | 0.3694 | **0.8000** |
| Outcome reward + VCC | **0.3262** | **0.3812** | 0.7800 |

Table 2: Ablation results for the VCC family. The pure commit reward did not improve rollout accuracy. The most useful variant kept the standard outcome reward and added the contrastive commit objective.

The ablations in Table 2 explain why our final VCC-RLOO row uses outcome reward plus contrast. Commit reward alone is intuitive, but it did not improve rollout accuracy. Although the commit-reward variants reached 0.8000 pass@16, their single-rollout accuracy was lower than the selected outcome+VCC-RLOO checkpoint. We believe this is because the scalar penalty adds another moving target to an already high-variance online RL objective. The contrastive loss is more local. It says that a verified clean commit should be preferred to the continuation that lost it, without replacing the outcome reward.

The later tight sweep did not improve the headline result. The selected validation checkpoint, coefficient 0.005 with seed 37 at step 60, reached only 0.2838 rollout accuracy and 0.7400 pass@16 on test. Other tight-sweep test checkpoints were closer, for example coefficient 0.0125 with seed 13 at step 60 reached 0.3150 rollout accuracy, but none exceeded the earlier outcome+VCC-RLOO coefficient 0.01 step 60 result. This negative result is still useful. It suggests that VCC-RLOO is sensitive to checkpoint timing and coefficient choice, and that validation on only 200 train prompts did not reliably select the best test checkpoint.

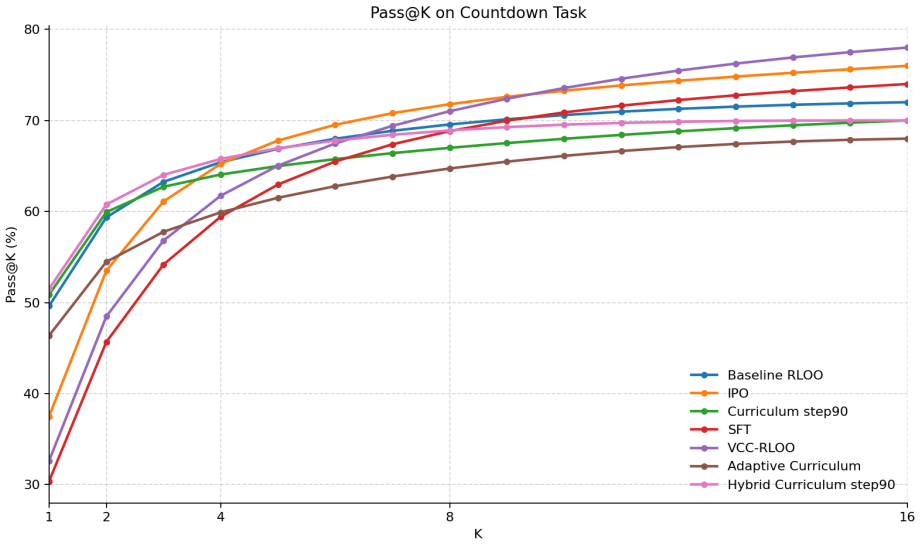


Figure 4: Pass@k on the held-out test set.

### 5.3 Qualitative Analysis

The most important qualitative finding is that answer abandonment is real and measurable. On the SFT eval, the model produced an average of 3.95 answer tags per rollout. Standard final-answer pass@1 was 0.3038, but first-verifying pass@1 was 0.3975. The gap is 0.0938, exactly 75 of 800 sampled rollouts. These are not fully wrong trajectories. They are trajectories where the model contained a correct answer somewhere, but final-answer scoring lost it.

One representative SFT rollout for target 98 and numbers $[44, 19, 35]$ first emits correct answers:

$$(44 + 19) + 35 = 98, \qquad (44 + 35) + 19 = 98.$$

Later, it emits

$$(35 + 19) + (44 - 35),$$

and incorrectly states that this equals 98. It actually equals 63, so the final verifier score is only 0.1. Standard RLOO treats this rollout as mostly wrong. VCC-RLOO turns it into a pair: the chosen response truncates at the first correct answer, and the rejected response is the later continuation.

For Curriculum RLOO, the qualitative pattern is different. The method does not inspect individual rollouts. It changes which prompts produce the rollouts. We observed the expected training dynamic: train rollout accuracy rose as the model trained on easy and medium prompts, then dropped when the active pool expanded to all prompts. We interpret this not as a collapse, but as the curriculum exposing the model to harder examples. The held-out test results support that interpretation: Curriculum step90 and the selected Hybrid Curriculum run both improved pass@1 over the RLOO baseline, though their lower pass@16 suggests reduced coverage.

## 6 Discussion

The two extensions helped in different ways. Curriculum RLOO is the cleaner single-sample performance result. The selected Hybrid Curriculum run improves pass@1 over the RLOO baseline, which means the average sample is slightly better. But its lower pass@16 suggests reduced exploration or less coverage across prompts. VCC-RLOO is the cleaner mechanism result. It directly addresses a failure mode visible in the data and improves pass@16 in the main comparison, but its single-rollout gain is modest.

The main limitation is scale. We evaluate on 50 held-out prompts with 16 responses each, so small differences should not be overinterpreted. The prompt-level standard errors are nontrivial. For example, VCC-RLOO improves pass@16 over SFT by 4 points, but both are estimated from only 50 prompts. We therefore present VCC-RLOO as a positive coverage result and a strong qualitative method, not as a statistically definitive win across all settings.

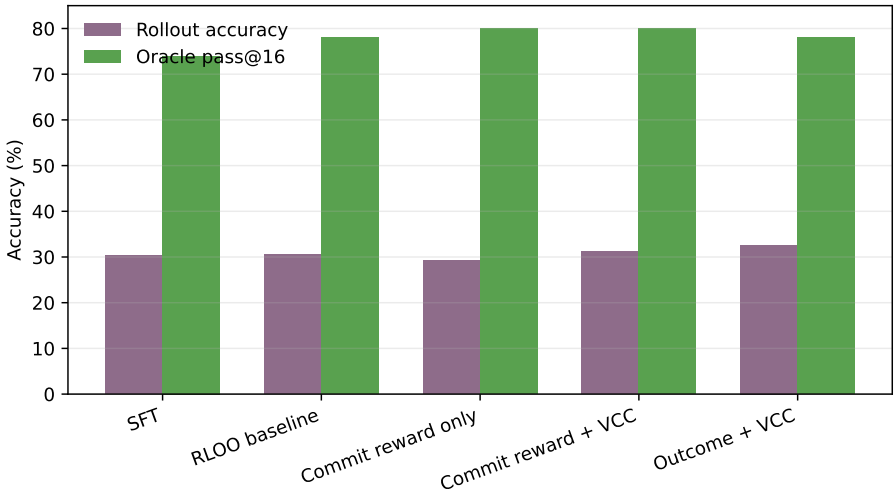


Figure 5: VCC ablation comparison. The best single-rollout VCC result uses outcome reward with a small verifier-generated contrastive auxiliary term.

There are also method limitations. The curriculum assumes that solution count is a good proxy for difficulty. That is reasonable for Countdown, but it is task-specific and may not transfer to domains where difficulty is not tied to the number of valid derivations. VCC-RLOO assumes that the verifier can identify a correct intermediate answer. It cannot help when the model never reaches a correct expression. It can also reward a correct final expression even if the surrounding chain of thought is confused, because the Countdown verifier checks the expression, not the reasoning trace. This is inherited from the task.

The broader impact is mostly about reliable small-model reasoning. Verifier-based RL can make compact models more capable on objective tasks, which is useful for lower-cost inference and educational tools. At the same time, any method that trains models to satisfy a verifier can overfit to the verifier. In this project that risk is contained by a toy arithmetic task, but the lesson matters: reward design and evaluation design are tightly coupled.

# 7 Conclusion

Our main takeaway is that sparse verifier rewards contain more structure than the final scalar reward reveals. Curriculum RLOO uses task structure before generation, selecting prompts likely to produce informative RLOO updates. VCC-RLOO uses rollout structure after generation, turning a verified but abandoned answer into a preference pair. On Countdown, the selected Hybrid Curriculum run gives the strongest single-sample policy, and the selected outcome+VCC-RLOO run gives the strongest pass@16 coverage in the main-method comparison. The most promising next experiment is to combine them: curriculum should increase the rate of rollouts with at least one correct answer, and VCC should teach the model to commit once it finds one.

# 8 Artificial Intelligence Disclosure

Per the course policies, we both have used AI for help (Chat GPT) in the writing of the report and code for the EXTENSION part of the final project.

# 9 Team Contributions

Zayn Malhotra led the SFT milestone and the original project setup, including supervised fine-tuning, the Countdown data pipeline, early evaluation, and proposal writing. For the final extension phase, Zayn implemented and ran the VCC family of methods, including commit-aware reward code, verifier-generated contrastive

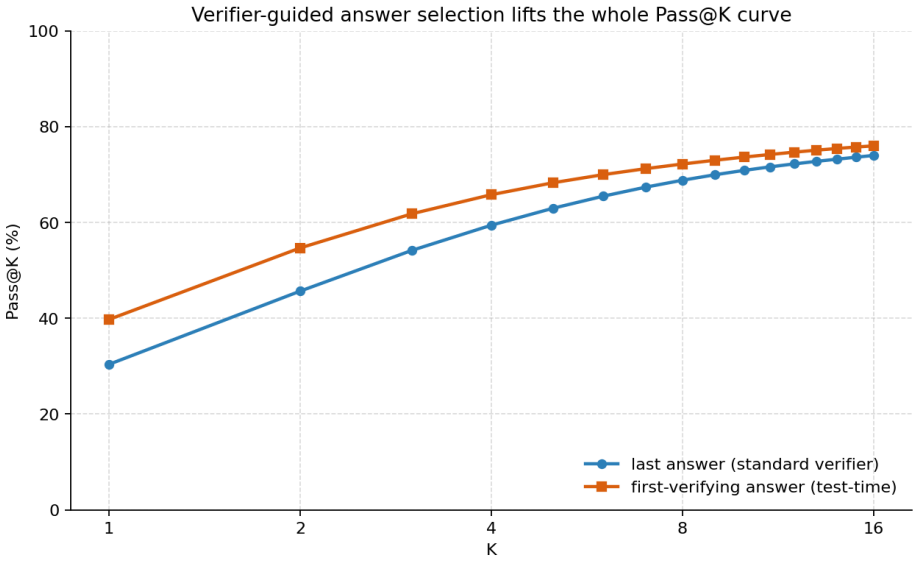


Figure 6: Diagnostic on SFT rollouts. If we select the first answer tag that verifies instead of the last answer tag, pass@k increases across the curve. This motivated VCC-RLOO.

pairs, Modal sweep launchers, checkpoint evaluation scripts, result summarization, and the tight VCC validation sweep. Zayn also edited the final report and helped produce the final poster narrative.

Ziyi Ding led major parts of the alignment milestone work, including IPO and baseline RLOO implementation, training, evaluation plots, and rollout analysis. For the final extension phase, Ziyi implemented Curriculum RLOO, including the solution-count difficulty computation, curriculum sampler, curriculum training runs, pass@k analysis, and presentation materials. Ziyi also contributed to the poster and will present the final project.

This contribution split differs from the proposal. The proposal planned a dense reward shaping extension, with Zayn focusing on the dense reward and Ziyi focusing on baseline RLOO and evaluation. After feedback that the proposed extension lacked novelty, we changed the final direction. The final project reports two more novel extensions, VCC-RLOO and Curriculum RLOO, both grounded in the observed failure modes of sparse-reward Countdown training.

# References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences, 2023. URL https://arxiv.org/abs/2310.12036.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

M. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/e57c6b956a6521b28495f2886ca0977a-Paper.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.