Extended Abstract

The project we choose for this report is the default course project. The default project studies the Countdown task [1], which evaluates the multi-step mathematical reasoning capabilities of language models. In this task, a model is given a set of numbers and a target value and must generate an arithmetic expression that uses the provided numbers to reach the target. The main research question of the default project is whether RL fine-tuning methods, such as Supervised Fine-Tuning (SFT), Preference Optimization (IPO), and Online Policy Gradient (RLOO), can improve multi-step mathematical reasoning in language models on Countdown. In our project extensions, we ask: how can we intelligently allocate inference-time and training-time compute to improve the success rate on the Countdown task? To this end, we investigate verifier-guided test-time inference in our first extension and curriculum learning for RLOO in our second extension.

Verifier-guided test-time inference. Our approach in the first extension is inspired by existing test-time compute research that samples multiple candidates to increase language model performance [2] and combines this with an explicit rule-based verifier for the Countdown task. We further investigate whether correct Countdown solutions are mostly concentrated in the model's likely completions or whether they can be better recovered through more exploratory sampling. In particular, we use verifier-guided best-of-k decoding as our inference-time strategy, where we freeze the base model, sample k candidate Countdown solutions, and use the exact verifier to select a correct solution if one appears. We further analyze the effect of sampling diversity in the k candidates by sweeping across various temperature settings.

We evaluate each of the SFT, IPO, and RLOO models trained in the default project with the above test-time inference strategy. For each model, we sample up to 32 completions per prompt and report pass@k for $k \in \{1, 2, 4, 8, 16, 32\}$. We further sweep across temperatures $T \in \{0.1, 0.3, 0.6, 1.0, 1.5\}$, where $T = 0.6$ is the default project's evaluation setting. Our results show that best-of-k decoding improves performance across all SFT, IPO, and RLOO models, while the base model without fine-tuning achieves near-zero accuracy across different k values. In addition, the sampling diversity across each k-candidate set, controlled by temperature, results in different performance trends for SFT, IPO, and RLOO, where SFT and IPO perform worse at high temperature while RLOO maintains high task accuracy and benefits more from exploratory sampling when k is large.

Curriculum learning for RLOO. In our second extension, we investigate how curriculum learning can be used during online RLOO training to improve learning under the same training budget. Motivated by prior work showing that providing easier samples before harder ones can improve the reward signal early in RL training and improve model performance [3], we investigate how different curriculum designs for the Countdown task affect model performance under the same training budget.

We explored multiple partitions of the training data samples, including partitioning based on the cardinality of the set of numbers used in the Countdown task and the type of operators used in the task. Under 30 steps of RLOO training, using data samples that only involve the operators "$+$" and "$-$" before moving on to samples with "$\times$" and "$\div$" results in the best final task performance, while the curriculum based on the cardinality of the number set only provides marginal improvement. We further show that under a limited step budget, such as the 30 steps we use, it is important to ensure each curriculum setting has a sufficient number of steps allocated for better training.

Limitations and conclusion. Overall, our results suggest that intelligently allocating both test-time and training-time compute makes a significant difference in the Countdown performance of language models. Best-of-k decoding helps when fine-tuning has already placed correct solutions in the model's output distribution but cannot create missing reasoning ability, and the degree of exploratory sampling varies across different RL fine-tuning algorithms. Curriculum learning improves model performance under the same training-time compute for RLOO, but it requires careful design of the curriculum itself. While our results are promising, the Countdown task has special properties such as an exact verifier, but many other reasoning tasks do not have such properties. Adaptive and less heuristic-dependent rules for choosing sampling k, sampling diversity, and the curriculum can further improve performance under similar compute budgets.

RL Fine-Tuning for Countdown Reasoning with Test-Time Verification and Curriculum Learning

Howard Xiao
Department of Electrical Engineering
Stanford University
howardx@stanford.edu

Weiwei Wu
Department of Electrical Engineering
Stanford University
wwwu@stanford.edu

Abstract

We study verifier-based RL fine-tuning for the Countdown arithmetic task, where a language model must generate an equation that uses each input number exactly once and reaches a target value. Building on the default SFT, IPO, and RLOO pipelines, we investigate two extensions for using test-time and training-time compute more effectively: verifier-guided test-time inference and curriculum learning for RLOO. For test-time inference, we use best-of-k decoding and temperature sweeps to study how generating multiple candidate solutions and exploratory sampling affect the performance of fine-tuned models. We find that larger k improves SFT, IPO, and RLOO. However, SFT and IPO degrade under high-temperature sampling, while RLOO is more compatible with exploratory decoding at larger k. For curriculum learning, we compare standard RLOO against easy-to-hard training data sampling under the same training compute budget, testing whether curriculum designs make sparse verifier rewards more informative. Overall, our results show that Countdown performance depends not only on the RL fine-tuning algorithm, but also on how inference-time and training-time compute are allocated.

1 Introduction

Large language models are increasingly used for tasks that require multi-step reasoning, including mathematical reasoning, code generation, and tool use. In these settings, a final answer can often be automatically checked even when the correct reasoning path is difficult to supervise directly. The Countdown task [1] is a controlled version of this problem: given a target number and a set of input numbers, the model must generate an arithmetic expression that uses each number exactly once and evaluates to the target value provided. Since each completion can be checked for format, number usage, and arithmetic correctness, Countdown provides a clean testbed for verifier-based reasoning [4].

We study the default post-training pipelines for Countdown using supervised fine-tuning (SFT), identity preference optimization (IPO), and REINFORCE leave-one-out (RLOO). These methods represent three different learning paradigms: imitation from expert demonstrations, offline learning from preference ranking, and online policy learning using verifier rewards. Our project focuses on how inference-time and training-time computation can be more intelligently allocated around these pipelines through verifier-guided test-time inference and through curriculum learning during training.

The first extension is to study whether verifier-guided best-of-k decoding, a strategy used at inference time, can reliably improve Countdown accuracy for fixed RL fine-tuned models trained with the above pipelines. Motivated by test-time compute scaling [2], we sample multiple candidate solutions and use the exact Countdown verifier to select a correct one if it appears. The hypothesis is that best-of-k decoding improves performance only when fine-tuning has already placed correct solutions with enough probability in the model's output distribution. We also study temperature as a sampling-

diversity control parameter, with the hypothesis that SFT and IPO are more sensitive to high-temperature sampling, while RLOO is more robust because it learns from sampled rollouts and verifier rewards.

The second extension is to study whether curriculum learning can improve RLOO under the same training budget. Online verifier-based RL can be inefficient when hard data samples produce mostly incorrect rollouts with little reward, resulting in a sparse reward signal early in training. Motivated by easy-to-hard RL for language-model reasoning [3], we test whether presenting easier Countdown samples before harder ones improves sample efficiency and investigate the effect of different curriculum designs. The hypothesis is that useful curricula increase the number of successful early rollouts, making RLOO updates more informative before training on harder Countdown cases.

Together, these extensions study a practical problem in verifier-based reasoning tasks: performance depends not only on the learned policy, but also on how inference-time and training-time compute are allocated. Our experiments evaluate how best-of-k sampling, temperature, and curriculum design interact with SFT, IPO, and RLOO on the Countdown task.

2 Related Work

Verifier-based RL fine-tuning for Countdown. The Countdown pipeline used in the default project uses SFT as a warm start for fine-tuning the base model before preference optimization with IPO and online RL with RLOO. SFT applies the standard next-token prediction objective only on completion tokens, so the model learns the response format and basic solution patterns from expert solution demonstrations [4]. IPO and RLOO then provide two different RL-based alignment signals: IPO learns from fixed chosen/rejected response pairs [5, 6], while RLOO learns from online sampled rollouts scored by a verifier reward [7]. In our work, these methods serve as base implementations and are not part of the contributions of our extensions.

Extension 1: Verifier-guided test-time inference. Prior work on test-time inference shows that language-model performance can improve by spending more computation during inference, for example by sampling more candidates or using search-like procedures [2]. Our test-time extension follows this direction, but in a more controlled verifier-based setting. Countdown gives an exact rule-based verifier, so we can directly test whether best-of-k decoding recovers correct solutions from a fixed post-trained model. Our key novelty is to compare how SFT, IPO, and RLOO respond to both larger k and different sampling temperatures in verifier-based best-of-k decoding.

Extension 2: Curriculum learning for RLOO. Curriculum learning studies whether models learn more efficiently when training examples are ordered from easier to harder. Prior work shows that easy-to-hard reinforcement learning can improve language-model reasoning by providing a more useful reward signal early in training [3]. Our curriculum extension applies this idea to Countdown RLOO under a fixed training budget. We compare difficulty definitions and different curriculum schedules for the task, testing which curriculum gives better final verifier-based performance than standard RLOO training under the same compute budget.

3 Method

3.1 Test-Time Inference

3.1.1 Verifier-guided best-of-k decoding

Our first extension considers an inference-time strategy for the Countdown task, where we keep the policy fixed after training with no model weight updates. Given a test Countdown prompt x, a trained policy π_θ uses verifier-guided best-of-k decoding by sampling k candidate completions

$$y_1, y_2, \ldots, y_k \sim \pi_\theta(\cdot \mid x).$$

Each completion is passed to the Countdown verifier $V(x, y)$, which returns 1.0 if the completion contains a correctly formatted equation that uses each input number exactly once and evaluates to the target, and returns 0.0 otherwise. We count the prompt as solved correctly if at least one sampled completion is verified to be correct.

This strategy is useful because it separates the post-trained model's capability from single-sample reliability. If pass@k is much higher than pass@1, then the model sometimes generates correct solutions, but its first sample is often unreliable. If pass@k remains low, then correct solutions are rarely present in the model's sampled outputs at all.

Algorithm 1 Verifier-Guided Best-of-k Decoding

Require: prompt x, trained policy π_θ, verifier V, sample count k
for $i = 1$ to k **do**
  Sample completion $y_i \sim \pi_\theta(\cdot \mid x)$
  Compute verifier score $s_i = V(x, y_i)$
  if $s_i = 1$ **then**
    return y_i, success
  end if
end for
return randomly chosen completion, failure

3.1.2 Temperature sweep for sampling diversity

The performance of the best-of-k decoding strategy depends not only on how many samples are drawn but also on how diverse those samples are. We control diversity using the sampling temperature T. At each decoding step, logits z are rescaled as

$$p_T(w \mid h) = \mathrm{softmax}(z(w)/T),$$

where h is the current context. A lower temperature concentrates probability on high-probability tokens and reduces sampling diversity, while a higher temperature gives lower-probability tokens a greater chance and increases sampling diversity.

We evaluate the same best-of-k procedure from Sec. 3.1.1 across multiple temperature settings. This tests whether correct solutions are concentrated near the model's most likely completions or spread into lower-probability regions. Our novelty relative to standard pass@k evaluation is that we use temperature as a diagnostic tool to compare the output distributions induced by SFT, IPO, and RLOO. In particular, we test whether offline preference optimization and online verifier-based RL respond differently to exploratory sampling during decoding.

3.2 Curriculum RLOO

Our second extension changes the Countdown sample distribution during RLOO training while keeping the verifier reward and total update budget fixed. Standard RLOO trains on the full Countdown distribution from the start of training. In contrast, our Curriculum RLOO first samples from easier data subsets and then shifts toward harder subsets. This tests whether curriculum RLOO can make sparse verifier rewards more useful. If early prompts are solvable more often, successful rollouts can provide a clearer learning signal before the model trains on harder cases.

We compare curricula based on two candidate difficulty definitions. The first partitions data prompts by the cardinality of the input numbers n, and the second partitions data samples by the required operator class. Because the input number count and operator class are correlated but not identical, we also evaluate a mixed curriculum that combines both signals. Across all variants, the objective, verifier, model, and 30-step RLOO update budget are unchanged, and only the subset of training data used at each stage varies.

4 Experimental Setup

4.1 Verifier-guided best-of-k decoding

We evaluate on the Countdown test set, where each prompt gives a target number and input numbers, and a completion is correct only if it uses every input number exactly once and evaluates to the target correctly. We compare four different models with test-time inference: the base pretrained model, SFT, IPO, and RLOO. The base model tests whether sampling alone can improve Countdown without

task-specific fine-tuning, while SFT, IPO, and RLOO test how verifier-guided decoding behaves after imitation learning, offline preference optimization, and online verifier-based RL.

For each model and prompt, we sample up to 32 completions and score every completion with the deterministic Countdown verifier. We report pass@k for $k \in \{1, 2, 4, 8, 16, 32\}$, where a prompt is solved correctly if at least one of the first k completions is verifier-correct. We also report exact sample accuracy, the fraction of individual completions that are verifier-correct. To study sampling diversity, we repeat the above pass@k and exact sample accuracy evaluations across temperatures $T \in \{0.1, 0.3, 0.6, 1.0, 1.5\}$, where $T = 0.6$ is the default evaluation setting in the default project milestones.

4.2 Curriculum RLOO

We compare the standard RLOO training algorithm used in the default project milestone against five curriculum schedules under the same 30-step RLOO update budget. Curriculum A trains for 15 steps on $n = 3$ Countdown prompts and 15 steps on $n = 4$ prompts. Curriculum B trains for 10 steps on $n = 3$ prompts and 20 steps on $n = 4$ prompts. Curriculum C trains for 10 steps on "$+$" or "$-$" prompts and 20 steps on "$*$" or "$\div$" prompts. Curriculum D trains for 10 steps each on "$+$" or "$-$", "$*$", and "$\div$" prompts. Curriculum E combines both difficulty definitions, using 5 steps on $n = 3$ prompts with "$+$" or "$-$", 5 steps on $n = 3$ prompts with "$*$" or "$\div$", 10 steps on $n = 4$ prompts with "$+$" or "$-$", and 10 steps on $n = 4$ prompts with "$*$" or "$\div$". Except for the curriculum schedule, in curriculum RLOO, every other hyperparameter is kept the same as the base RLOO training in the project milestone.

5 Results

5.1 Verifier-Guided Best-of-k Decoding

5.1.1 Quantitative Evaluation

Figure 1 and Figure 2 show pass@k as the number of sampled completions increases. The base model is omitted in these figures because its pass@k stays near zero across settings, showing that sampling more does not create Countdown ability by itself. After fine-tuning, SFT, IPO, and RLOO all improve as k increases. For example, at the default temperature $T = 0.6$, SFT improves from pass@$1 = 0.27$ to pass@$32 = 0.80$, IPO improves from 0.35 to 0.82, and RLOO improves from 0.50 to 0.70. This shows that verifier-guided best-of-k decoding can trade inference-time compute for higher success probability once fine-tuning has placed correct solutions in the model's output distribution.

Temperature has different effects across models trained with different post-training methods. SFT and IPO are much worse at very high temperature where the sampling diversity is large. At $T = 1.5$, SFT reaches only pass@$32 = 0.68$, compared with 0.84 at $T = 1.0$; IPO also reaches only pass@$32 = 0.68$, compared with 0.84 at $T = 1.0$. This suggests that their correct solutions are mostly near high-probability completions, and high-temperature sampling adds a lot of low-quality completions that hurt performance.

However, at high temperature, the model post-trained with RLOO behaves differently. At small k, high temperature is worse: the RLOO model's pass@1 drops from about 0.50 at $T = 0.6$ to 0.36 at $T = 1.5$. But at larger k, high-temperature sampling becomes useful: the RLOO model reaches pass@$32 = 0.78$ at $T = 1.5$, higher than its $T = 0.6$ pass@32 of 0.70. Figure 3 further clarifies this effect. RLOO has the highest exact sample accuracy overall, around 0.48–0.50 at 32 samples for moderate temperatures, while SFT and IPO are closer to 0.26–0.35. Thus, RLOO produces individually correct samples more often; therefore, its high-temperature samples remain useful enough for the verifier to recover correct answers when k is large.

5.1.2 Qualitative Analysis

The difference between the performance of models post-trained with RL algorithms IPO and RLOO can be further explained using their training signals. IPO is trained on fixed offline chosen or rejected answer paths. For each prompt, the update directly makes the chosen token sequence more likely than the rejected token sequence. This improves the model near the preferred answer paths in the dataset,

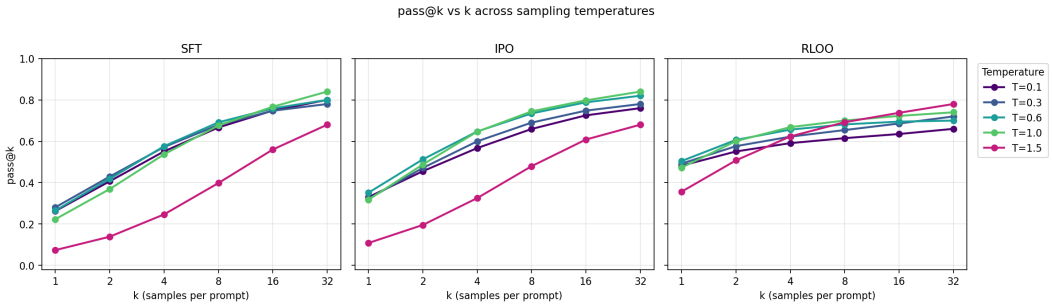

Figure 1: Pass@k as a function of the number of sampled completions for SFT, IPO, and RLOO across temperatures.

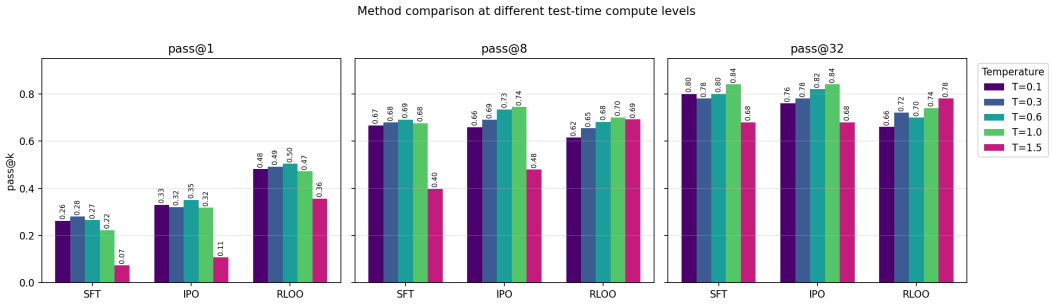

Figure 2: Method comparison at pass@1, pass@8, and pass@32 across sampling temperatures.

but gives little direct feedback on alternative token paths that only appear under high-temperature sampling. Therefore, when temperature is very high, IPO often explores regions where its completions are less reliable.

In contrast, RLOO is trained from the model's own sampled rollouts. During training, the model generates completions, the verifier scores them, and successful sampled paths are reinforced. This means RLOO receives feedback on the kinds of outputs it actually produces, not only on fixed offline examples. This helps explain why RLOO is more compatible with exploratory best-of-k decoding: a single high-temperature rollout is risky, but many diverse rollouts can expose additional potentially correct solutions for the verifier to select. This analysis also explains why pass@k and exact sample accuracy are both required for our analysis. Pass@k measures whether a correct answer can be recovered from a set of samples, while exact sample accuracy measures how often individual samples are correct. RLOO performs well on both while the IPO model's performance improves with k but decreases in high-temperature settings as high-temperature exploration mostly adds only low-quality samples. Therefore, the IPO-trained model's best performance comes from moderate-temperature decoding while RLOO-trained models can use a higher temperature during decoding in test-time inference at large k.

5.2 Curriculum RLOO

5.2.1 Quantitative Evaluation

Figure 4 shows that defining Countdown difficulty does not depend only on how many numbers appear in the question prompt. Prompts with $n = 4$ can be harder for baseline RLOO, but they also contain more questions that use multiplication and division than $n = 3$ prompts. This makes the difficulty split ambiguous: the model may struggle because there are more numbers in the question prompt, because the required operations are harder, or because both factors interact. We therefore test curricula that partition prompts by both number count and operator class.

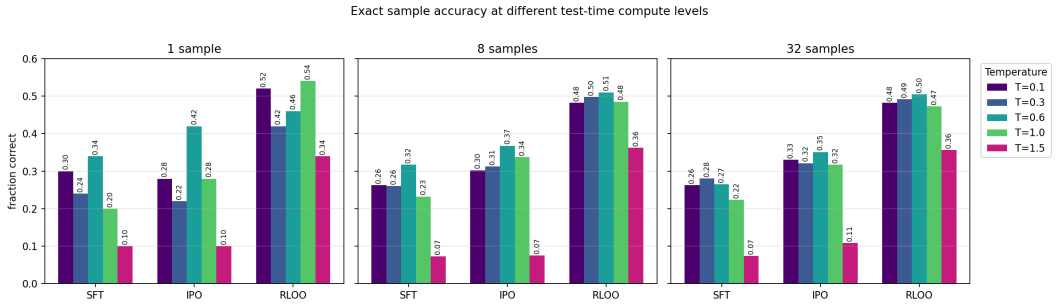

Figure 3: Exact sample accuracy, measured as the fraction of sampled completions that pass the verifier.

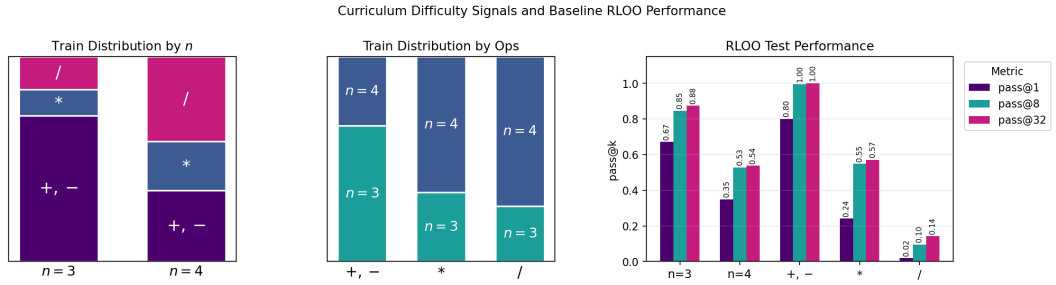

Figure 4: Potential curriculum difficulty partitions for Countdown prompts.

Table 1 and Figure 5 show that Curriculum D is the strongest among the RLOO variants, reaching pass@1 $= 0.53$, pass@8 $= 0.73$, and pass@32 $= 0.80$ under the same 30 training steps, higher than other curricula and also the base RLOO-trained model from the project milestone. This suggests that operator-aware curriculum design is more effective than number-count curriculum alone. In contrast, Curriculum A and B, which split the training compute budget only by $n = 3$ and $n = 4$ prompts, give less improvement over standard RLOO training. Curriculum B improves performance more than Curriculum A, suggesting that spending enough steps on the harder partition matters, but the best gains come from curricula that explicitly separate $+/-$ from multiplication/division cases as shown by the results in Curriculum C and Curriculum D. Since the compute budget is limited to 30 update steps for RLOO, Curriculum E does not improve performance as much as Curriculum C and D, suggesting that the granularity of curriculum design needs to take into account the total training budget, as having few steps for each curriculum partition may cause learning instabilities.

5.2.2 Qualitative Analysis

Figure 6 further shows that the easiest subsets are already close to saturated across all curricula, so the useful gains come from broader coverage of the hard arithmetic tail. This explains the improved performance of Curriculum B compared to Curriculum A. In particular, $n = 3$ and "$+$"/"$-$" prompts leave less room for improvement, while $n = 4$ and hard "$*$" or "$/$" cases expose the sparse-reward problem more clearly. The curriculum takeaway is that easy stages should be chosen not to spend limited RLOO steps over-training on already-solved prompts, but instead to prepare the policy for harder operator-driven cases in the Countdown task.

6 Discussion

Our results show that verifier-guided decoding and curriculum learning can use compute more effectively and improve Countdown performance, but both extensions have clear limitations. For test-time inference, a fixed k may waste compute: easy prompts may need only one or two samples,

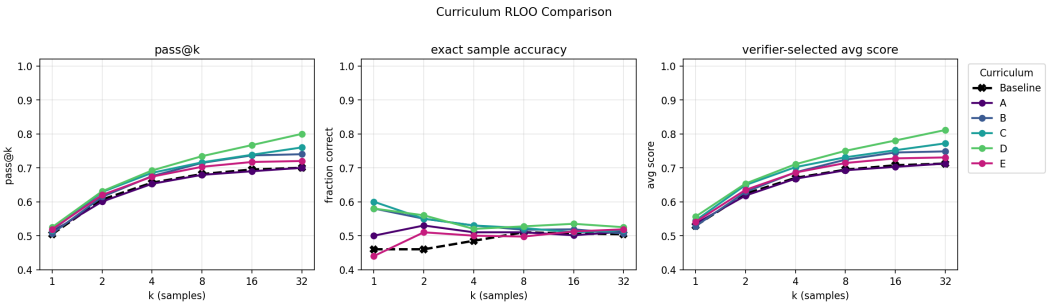

Figure 5: Curriculum comparison across pass@k metrics under the same 30-step RLOO budget.

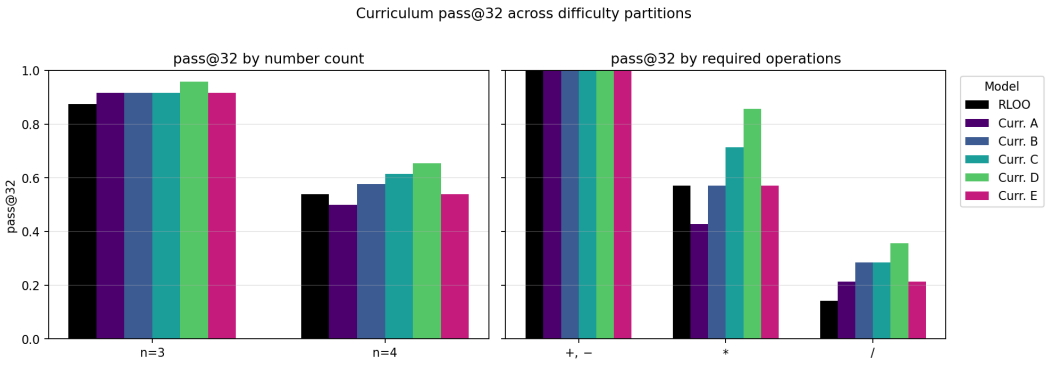

Figure 6: Pass@32 by difficulty split for standard and curriculum RLOO variants.

while hard prompts may need many more. A more efficient system should stop sampling once the verifier finds a correct answer and allocate extra samples only to prompts that remain unsolved.

Temperature can also be model-specific. IPO and SFT perform best when sampling stays closer to high-probability completions, while RLOO can benefit from more exploratory decoding when k is large. This means a single default temperature is unlikely to be optimal across training methods. In future work, k and temperature should be selected jointly, because high temperature is harmful when only one or two samples are drawn but can help when the verifier can choose from many diverse completions.

For curriculum learning, the main difficulty is defining data sample difficulty effectively. Fewer input numbers or smaller targets do not always mean a prompt is easier for the model. Operator type, number count, and empirical rollout success can give different difficulty rankings. This makes hand-designed curricula less robust. Future curriculum schedules should adapt to online rollout accuracy rather than relying only on fixed labels.

More broadly, our findings suggest that smarter allocation of test-time and training-time compute can lower the cost and energy footprint of deploying reasoning models, while making already-trained more performant under constrained budgets. Because the verifier-guided and curriculum strategies we study extend naturally to any domain with verifiable answers, such as code generation and theorem proving, they offer a practical path toward more reliable and resource-efficient reasoning systems.

7 Conclusion

Overall, our results suggest that intelligently allocating both test-time and training-time compute makes a significant difference in the Countdown performance of language models. Best-of-k decoding helps when fine-tuning has already placed correct solutions in the model's output distribution but cannot create missing reasoning ability, and the degree of exploratory sampling varies across different

Table 1: Curriculum comparison under the same update budget.

Name	pass@1	pass@4	pass@8	pass@16	pass@32
SFT	0.27	0.58	0.69	0.76	0.80
IPO	0.35	0.65	**0.73**	**0.79**	**0.82**
RLOO	0.50	0.66	0.68	0.69	0.70
Curr. A	0.51	0.65	0.68	0.69	0.70
Curr. B	0.51	0.68	0.71	0.74	0.74
Curr. C	0.52	**0.69**	0.72	0.74	0.76
Curr. D	**0.53**	**0.69**	**0.73**	0.77	0.80
Curr. E	0.52	0.67	0.70	0.72	0.72

RL fine-tuning algorithms. Curriculum learning improves model performance under the same training-time compute for RLOO, but it requires careful design of the curriculum itself. While our results are promising, the Countdown task has special properties such as an exact verifier, but many other reasoning tasks do not have such properties. Adaptive and less heuristic-dependent rules for choosing sampling k, sampling diversity, and the curriculum can further improve performance under similar compute budgets. Our results show that, for verifier-compatible reasoning tasks, how compute is allocated at inference and training time can matter as much as which fine-tuning algorithm is used for post-training.

8 Team Contributions

- **Group Member 1: Howard Xiao** Howard implemented the Project Milestones together with Weiwei Wu and wrote project milestone reports together. Howard implemented and tested the curriculum learning extension and produced results for the extension. Howard and Weiwei created the Final Project poster and report together.
- **Group Member 2: Weiwei Wu** Weiwei implemented the Project Milestones together with Howard Xiao and wrote project milestone reports together. Weiwei implemented and tested the test-time inference extension and produced results for the extension. Weiwei and Howard created the Final Project poster and report together.

Changes from Proposal Our chosen extensions, implementations, and result analysis match what we proposed in the project proposal.

References

[1] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

[2] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024.

[3] Shubham Parashar et al. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning, 2025.

[4] CS224R Course Staff. Rl fine-tuning of language models: Default project guidelines, 2026.

[5] 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.

[6] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[7] 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, 2024.