

Extended Abstract

Motivation Reinforcement learning techniques quickly became state of the art for fine-tuning / post training of LLMs. PPO and DPO approaches - performant algorithms of the time - were used to align model output with human preferred task completions. However, these types of algorithms require *multiple* instances of the model at training time to produce differing outputs. For LLMs, this required mass amounts of memory and compute. The next generation of techniques, GRPO and RLOO, use a single LLM to produce *groups* of responses from which advantages are calculated during training time. Although these algorithms greatly reduce the resources required to post train LLMs, for large groups, computing the statistics for each training iteration across batches can be both time consuming and resource intensive. In this paper, we liken the group advantage within these approaches to adversarial games from MARL literature and provide simplified advantage calculations for rewards structured in a zero-sum game formulation. Ultimately, we show that this formulation has the potential to reduce the compute power required during training for group-based LLM training.

Method Theoretical: To our knowledge, no advantage calculation simplification has been done based on zero-sum game frameworks. As motivation, we first looked at the advantage calculations for the group based approaches. For RLOO: $\hat{A}_i^{RLOO} = R_i - \frac{1}{N-1} \sum_{j \neq i} R_j$; and for GRPO: $\hat{A}_i^{GRPO} = \frac{R_i - \text{mean}(\vec{R})}{\text{std}(\vec{R})}$. We then apply the zero-sum game property $\sum_{i=1}^N R_i = 0$ to derive group statistics and simplify these advantages into $\hat{A}_i^{RLOO} = R_i + \frac{R_i}{N-1}$ and $\hat{A}_i^{GRPO} = N^{\frac{1}{2}} \frac{\vec{R}}{\|\vec{R}\|_2^2}$.

Empirical: We evaluate the theoretical results by making a few key changes: (1) we first reformulate the reward structure of the countdown scoring approach to adhere to the zero-sum property; (2) we modify the advantage calculations for RLOO to be the simpler version derived in the *Theoretical* section; and (3) we perform an ablation study comparing the vanilla RLOO and varying one component at a time, building up to the full Zero-Sum Advantage formulation.

Implementation We implemented two new reward functions. Both center the sum or rewards around zero, but only one takes into consideration the length to get to the answer. We accomplish (1) from above by comparing groups of two responses. If both responses get a correct solution, the one that gets to the solution in less tokens is rewarded while the slower response is penalized in equal magnitude. If one response is correct while the other is incorrect, a +1 / -1 reward is given to each respectively. If one response formatted correctly while another is incorrectly, a +0.1 / -0.1 reward is given respectively. A reward of 0 is given to both otherwise. The advantage implementation builds upon RLOO and calculates the advantage using the simplified formulas above.

Results Our results indicate that the zero-sum advantage simplification is less *sample efficient* than RLOO; however it has promising *compute-time* benefits for larger groups, a subject touched upon in the future work section. The 200 step zero-sum approach beats the 100 step RLOO for the first six Pass@K values of k. Additionally, the zero-sum advantage trained models' capability drops off for Pass@k for larger K - asymptotically approaches 70% while RLOO exhibits no such bound.

Discussion All of the current results are from small sample sizes and start from the same, less capable, SFT base model due to limited compute budget. Running our study on a more capable model could showcase even more interesting results as it would receive more dense 'conciseness' rewards. Additionally, the default RLOO implementation's rewards are *very* close to zero for most samples. It would be interesting to measure the improvement of our zero sum implementation to a problem that does not already have a mean-dependent term $\frac{\mu N}{N-1}$ that could be lost in the noise. Finally, groups of only up to eight were used for RLOO and Zero-Sum training; as we expect Zero-Sum to exhibit faster training for larger groups, it would be worthwhile to measure the compute time required per Pass@K performance for a growing number of group sizes.

Conclusion Our zero-sum theoretical approach is sound and shows promise empirically in *large group* wall-clock time training time reduction. Despite a reduced sample efficiency and what appears to be an asymptotic bound on performance, this approach should be evaluated across different tasks with a larger group size to tease out where the wall-clock time benefit outpaces the current limitations and where performance limitations appear within the zero-sum game reformulations.

Zero Sum Game Framework for Group Advantage Simplification

Joshua Slagle

Department of Computer Science
Stanford University
jstagle@stanford.edu

Abstract

While advances in large language model training techniques have allowed more performant models to be trained with less compute during training time, performant LLM training is still prohibitively expensive for many organizations. To combat this, we highlight parallels between group-based advantage estimation techniques and adversarial self-play in reinforcement learning, and use the zero-sum game formulation to derive simplified expressions for state-of-the-art (SOTA) techniques' group advantage calculations. We then create a zero-sum formulation for the popular *countdown* task for LLMs and use it as a representative task to explore how these simplified expressions work in practice. Our empirical results showcase the validity of our theoretical derivations, and hint at significant train-time compute savings for training parameters that use large groups. Given these results, our zero-sum advantage formulation allows for more efficient training of LLMs while simultaneously enabling larger group sizes to be used for baseline estimation, allowing for more stable baseline estimation with less variance.

1 Introduction

Large language models have achieved widespread adoption in part because of their ability to help with - and complete - tasks as assigned to them via prompting. It comes as no surprise then, that reinforcement learning quickly became the go-to training paradigm for large language model task performance optimization. However, reinforcement learning, in general, is not extremely sample or compute efficient. Initially, PPO Schulman et al. (2017) and DPO Rafailov et al. (2024) algorithms were used as state of the art LLM training techniques; however, these approaches required *multiple* instances of the model to be instantiated, levying enormous memory and compute requirements for LLM training. In the next generation of approaches, algorithms like GRPO and RLOO use a *single* LLM to create a group of responses from which baselines are calculated.

While this new generation of approaches save the memory and compute related to hosting multiple large language models simultaneously, they still require statistics to be calculated over the entire group for every training update / mini-batch sample. For large groups, these re-calculations of group statistics for every cycle can make up a sizable slice of the overall training time compute.

To address this inefficiency, we investigate the parallels between group advantages and adversarial games in multi-agent reinforcement learning, and leverage reward structuring techniques within that space to derive simplified group advantage calculations that use less compute. Of most emphasis is the theoretical application of a zero-sum game structure to the general form of group advantage expressions. The theoretical derivations are then followed by the simplified advantages practical evaluation for RLOO as applied to the countdown task with reformulated zero-sum game reward structure.

The Countdown Task: The countdown task is a problem formulation where an agent is provided, as input, a small - typically three or four - set of integers and a target integer, and as output, expects a set of operations $+$, $-$, \times , \div that map the input integers into the target number.

Group Advantage Estimation: Group advantage estimation techniques use a sampled set of responses from an LLM from the same prompt to calculate statistics (mean, standard deviation, etc) that are then used later as a part of a baseline function for that prompt. As a most basic example, the advantage of a single response i from a group of size G responses would be

$$A_i^{simple} = R_i - \sum_{k=0}^{G-1} R_k$$

where A_i^{simple} is a measure of how much better response i was as compared to the entire group of responses.

Adversarial Self Play: Adversarial self-play in reinforcement learning is a training method in which an agent improves by competing against exact - or only slightly modified - copies of itself and its own strategy. Adversarial self-play acts as a feedback loop: as the agent’s policy becomes more performant, so does its adversary, ensuring that the model continually improves. Adversarial self-play is extremely popular in game playing as demonstrated in Go Silver et al. (2017) and StarCraftII Vinyals et al. (2019) and LLM reasoning and alignment as in SPAG Cheng et al. (2025).

Advantage Estimation and Adversarial Self Play Parallels: In general, adversarial self-play is a framework where an agent competes against itself for rewards. This can take many forms, however the common theme is that an increase in one agent’s rewards corresponds to a decrease in one or more competing agents’ rewards; notably, these increases and decreases do not need to be of the same magnitude. The only constraint is that the reward of one agent is negatively correlated with a function of the rewards of all other agents

This same negative correlation appears in the advantage calculation where the advantage of agent i , A_i , decreases as the other agents’ rewards increase. For example, for a given agent $j \neq i$:

$$\begin{aligned} \frac{\delta}{\delta R_j} A_i^{simple} &= \frac{\delta}{\delta R_j} \left(R_i - \sum_{k=0}^{G-1} R_k \right) \\ &= \frac{\delta}{\delta R_j} \left(R_i \right) + \frac{\delta}{\delta R_j} \left(- \sum_{k=0}^{G-1} R_k \right) \\ &= 0 + -1 = -1 \leq 0 \end{aligned}$$

So as R_j decreases, R_i increases and vice versa.

Zero-Sum Game Formulation: If the increases and decreases mentioned in the ‘Advantage Estimation and Adversarial Self Play Parallels’ section *are guaranteed* to always have the same magnitude as averaged across all agents, we call this a *fixed-sum* game.

$$\sum_{i=0}^{N-1} R_i = C$$

where N is the number of competing agents, R_i is the reward from agent i , and C is some constant. In the special case where $C = 0$, we get the definition of a *zero-sum* game:

$$\sum_{i=0}^{N-1} R_i = 0$$

More specifically, our contributions in this paper are

- Alignment of group advantage estimates with adversarial multi-agent reinforcement learning
- Practical guidelines of how to reformulate LLM task rewards within a zero-sum game framework, using the countdown task as an example

- Theoretical results of applying zero-sum reward structure to group advantage estimate calculations and the derivation of simplified expressions for efficient compute
- Empirical results of examining the simplified group advantages in RLOO as evidence for approach soundness

2 Related Work

Our work applies concepts from and adversarial multi-agent reinforcement learning and game theory to the LLM group advantage estimation techniques use for post-training. As such, we these types of problems have been looked at from different angles from homes in different fields.

2.1 Group Advantage Estimation

Within the general group advantage estimation, multiple techniques have been devised that focus on different estimation methods. Two of the most performant approaches are REINFORCE Leave One Out (RLOO) Ahmadian et al. (2024) and Group Relative Policy Optimization (GRPO) Shao et al. (2024). Both approaches utilize group statistics (mean, standard deviation, etc) calculated on the fly to build up a baseline function for advantage calculation of each response.

2.1.1 RLOO Advantage

RLOO specifically focuses on sub-group statistics. That is, RLOO generates a set of responses from the same prompt, and uses these responses to calculate a group-based advantage by subtracting the average of the group with the reward in focus for the advantage calculation removed from the group (leaving it out). In summary, the advantage function within RLOO answers the question, "How much better is my chosen response than the average of all *other* responses?" Mathematically, this is represented as follows:

$$\hat{A}_i = R_i - \frac{1}{N-1} \sum_{j \neq i} R_j$$

and this advantage is used to weight updates to the currently acting policy.

2.1.2 GRPO Advantage

GRPO is a similar group-based advantage estimator, but instead of asking the question "How much better is my chosen response than the average of all *other* responses?", it asks, "How much better is my chosen response than the average response from this group?". In doing so, it reduces the need to compute a new advantage baseline for each reward and simplifies the calculation into subtracting the mean and dividing by the standard deviation for each reward. Symbolically, this is:

$$\hat{A}_i = \frac{R_i - \text{mean}(\vec{R})}{\text{std}(\vec{R})}$$

2.2 Multi-Agent Adversarial Self Play in LLMs

Co-evolving LLM agents in tandem with each other is not a new idea. Different instantiations of this problem have been explored with agents taking on different roles, such as those provided in the project assignment (1) Dong and Ma (2025) and Subramaniam et al. (2025). In Dong and Ma (2025) the focus is on using self-play on two agent types - a conjecturer and a prover - to generate high quality data, essentially using the self-play formalism with rewards as an intelligent problem space search. Subramaniam et al. (2025) uses multiple agents during training, but in a non-standard multi-agent formalism and with the sole goal of generating a more diverse dataset. Others use adversarial multi-agent frameworks closer to their pre-llm formalisms such as Zhu et al. (2026) and have co-evolving agents such as in Ma et al. (2024). Impressively, Zhu et al. (2026) uses true multi-agent and game theoretic metrics to measure the performance gains in a competitive setting, showing that the agents can reach nash-equilibriums.

2.3 Zero-Sum Games

Zero-sum games are frameworks where the rewards of competing policies always sum to zero for a given, closed and finite set of interactions within a 'game'. This defining property manifests as the below, where N is the number of competing policy within the game and R_i is the reward for a single policy i :

$$\sum_{i=1}^N R_i = 0$$

Zero sum games are critical to proper self-play, yet current literature does not formulate the games with a reward structure that strictly adheres to this requirement such as in Dong and Ma (2025) and Ma et al. (2024). Similarly, we can make parallels between advantages and *local* zero sum game formulations, and make connections to GRPO Shao et al. (2024) where the agent is competing against itself, albeit implicitly through zero-mean advantage functions. Other multi-agent formulations have different explicit roles for each goal Subramaniam et al. (2025) which increases the number of inference passes required during training; here, we use the zero-sum self-play game formulation for the following reasons, with the last being the emphasis of this paper:

- **Guaranteed existence of stable solution** known as the Nash Equilibrium
- **Self-Play** generates robust strategies that cannot be beaten within game rules
- **Zero mean rewards** that enable algorithms to greatly simplify the advantage calculations over groups of agent responses

While we see group advantages used to estimated with group-based statistics in RLOO and GRPO, multi-agent adversarial frameworks used to generate diverse dataset and reach nash equilibriums, and zero-sum frameworks used in famous results such as AlphaZero and SPAG we do not see zero-sum game frameworks applied to LLM tasks to simplify advantage calculations at training time.

3 Method

3.1 Extension Description Overview

This section breaks down our contributions into the theoretical derivation contributions and the implementaiton / practical guidance contributions. Within the theoretical section, we formulate the countdown task as a zero-sum game, and then use the defining property of the zero-sum game framework to greatly simplify the GRPO and RLOO advantage estimates, enabling training time compute reductions. In the implementation section, we outline the framework in which we investigate the practicality and legitimacy of these derivations when applied to the countdown task.

3.2 Theoretical Contributions

3.2.1 Zero-Sum Game Formulation of The Countdown Task

In order to leverage the zero-sum game formulation's properties the promised group advantage estimation, we must first transform the countdown task into a zero-sum game. We show this specifically for 1v1 comparison games for simplicity and compute credit constraints, but the approach is easily extended to groups of all sizes as shown in 1. This is done by, first, extending the group sampling method to sample - in our case - twice, with group sizes cut into half. As an example, given a single prompt x and group size 8, we pull one set of sample responses y_1 of size 4 from our LLM, immediately followed by another set of sample responses y_2 of size 4 from the same LLM. Each element of y_1 and y_2 are then scored against the other in the same position using the Zero-Sum Oracle Scorer shown in 1. (that is $y_1[0]$ gets compared with $y_2[0]$). The scoring adheres to the following rules to ensure zero-sum rewards:

- If both responses get a correct solution, the one that gets to the solution faster - in number of tokens - is rewarded with respect to how concisely the solution was found while the slower response is penalized in equal magnitude
- If one response is correct while the other is incorrect, a +1 / -1 reward is given to each respectively

- If one response formatted correctly while another is incorrectly, a +0.1 / -0.1 reward is given respectively
- If both responses are wrong or formatted incorrectly, a reward of 0 is given to each

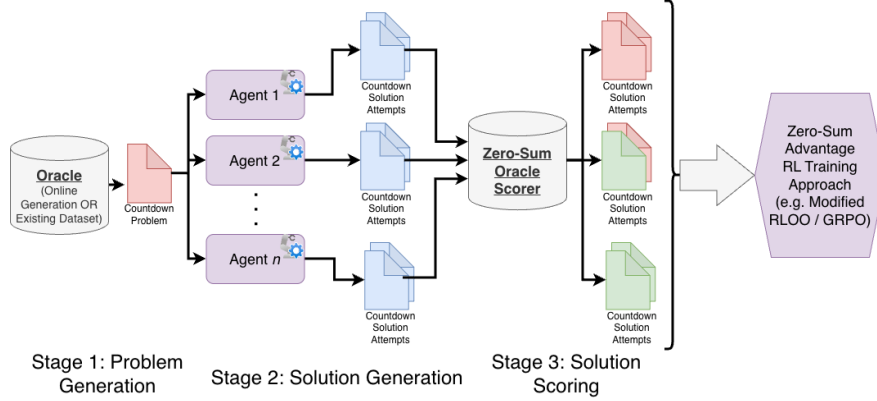


Figure 1: Method Overview.

3.2.2 Group Advantage Simplification Mathematics

Given the properties of our zero-sum game rewards, we can greatly simplify group-based advantage functions. We start by using the defining property of a zero-sum game

$$\sum_{i=1}^N R_i = 0$$

to derive the full-group statistics

$$mean(\vec{R}) = 0 \quad \left| \quad std(\vec{R}) = \left(\frac{\sum_{i=1}^N (R_i^2)}{N} \right)^{\frac{1}{2}}$$

where N is the number of competing responses and R_i is the reward for a single response i .

GRPO: We start with GRPO, as the simplification almost directly falls out of the above:

$$\begin{aligned} \hat{A}_i^{GRPO} &= \frac{R_i - mean(\vec{R})}{std(\vec{R})} \\ &= \frac{N^{\frac{1}{2}}(R_i - 0)}{\sum_{i=1}^N (R_i^2)} \\ &= N^{\frac{1}{2}} \frac{R_i}{\sum_{i=1}^N (R_i^2)} \\ \Rightarrow \hat{A}^{GRPO} &= N^{\frac{1}{2}} \frac{\vec{R}}{\|\vec{R}\|_2^2} \end{aligned}$$

where in the last line, we have the simplified form of the entire advantage function tensor rather than a single advantage calculation. Of note, this representation can leverage the compute-efficient implementations of vector normalization.

RLOO: For RLOO, this process takes a bit more manipulation. We first must draw from the definition of the mean of a dataset with one point removed:

$$\mu' - \mu = \frac{\mu - R_i}{N - 1}$$

where μ' is the mean of the subgroup with one datapoint removed, μ is the mean of the entire group, R_i is the reward of the one data point removed, and N is the number of responses within a group.

Conceptually, μ' is the same as our RLOO baseline term $\frac{\sum_{j \neq i} R_j}{N-1}$; so we are motivated to manipulate the above equation into something with only μ' on the left hand side.

$$\begin{aligned}\mu' - \mu &= \frac{\mu - R_i}{N - 1} \\ \mu' &= \frac{\mu - R_i + (N - 1)\mu}{N - 1} \\ \mu' &= \frac{(1 + N - 1)\mu - R_i}{N - 1} \\ \mu' &= \frac{N\mu}{N - 1} + \frac{-R_i}{N - 1}\end{aligned}$$

And thus our baseline RLOO term is equal to:

$$\frac{\sum_{j \neq i} R_j}{N - 1} = \mu' = \frac{N\mu}{N - 1} + \frac{-R_i}{N - 1}$$

Given our zero-sum game property of $\mu = 0$, we get

$$\frac{\sum_{j \neq i} R_j}{N - 1} = \frac{-R_i}{N - 1}$$

Plugging this into our RLOO advantage from before:

$$\hat{A}^{RLOO}_i = R_i + \frac{R_i}{N - 1}$$

which now relies only on the reward received for the single response action during that interaction. This result is our key contribution from a theoretical perspective, and the primary area of focus in the experimental portion. To our knowledge, no other work has been done in simplifying group-based advantage estimation calculations through targeted reward structuring such as within a zero-sum game framework.

The zero-sum formulation of the countdown game is simply a means to get to this simplified advantage form. In our Zero-Sum formulation of RLOO then, we replace the group-based advantage calculation portion with the above derived result and evaluate the training's effectiveness. In order to use this simplified advantage for RLOO, zero-sum reward structures are *required*, and thus we assume that the task we are applying RLOO to either has, or is capable of supporting a reformulation into, a zero-sum game framework.

3.3 Implementation

We evaluate the theoretical results by making a few key changes: (1) we first reformulate the reward structure of the countdown scoring approach to adhere to the zero-sum property following the process outlined in 3.2.1 . (2) we modify the advantage calculations for RLOO to be the simpler version derived in the 3.2.2; and (3) we perform an ablation study comparing the vanilla RLOO and varying one component at a time, building up to the full Zero-Sum Advantage formulation.

3.3.1 Implementation Contributions

We implemented two new reward functions as a part of our implementation contributions, both of which follow the zero-sum game framework which can be used for further work on zero-sum countdown tasks. The first is the new length-based zero-sum reward function as one primary contribution, while the second is a zero-sum reward function that does not take into consider length for the ablation study. We accomplish (1) from the theoretical paragraph by comparing groups of two responses. If both responses get a correct solution, the one that gets to the solution in less tokens is rewarded with respect to how concisely the solution was found while the slower response is penalized in equal magnitude. If one response is correct while the other is incorrect, a +1 / -1 reward is given to each

respectively. If one response formatted correctly while another is incorrectly, a +0.1 / -0.1 reward is given respectively. If both responses are wrong or formatted incorrectly, a reward of 0 is given to each.

SFT Starting Point

We use, as the baseline model from which we train all our ablation branches off of, the supervised fine tuning (SFT) model that was a part of the original default project, that optimized the following objective on the provided preference dataset.

$$\max_{\theta} \mathbb{E}_{x,y \in D} \sum_{t=1}^{|y|} \log \pi_{\theta}(y_t | x, y_{<t})$$

RLOO Baseline

We then use the RLOO objective function to train our comparison baseline model:

$$\frac{1}{k} \sum_{i=1}^k \left[R(y_{(i)}) - \frac{1}{k-1} \sum_{j \neq i} R(y_{(j)}, x) \right] \nabla \log \pi(y_{(i)} | x) \quad \text{for } y_{(1)}, \dots, y_{(k)} \sim^{i.i.d.} \pi_{\theta}(\cdot | x)$$

RLOO with Simplified Advantage

and as a part of our contribution implement RLOO with the updated training loop that uses the simplified RLOO advantage function in the optimization objective:

$$\frac{1}{k} \sum_{i=1}^k \left[R_i + \frac{R_i}{k-1} \right] \nabla \log \pi(y_{(i)} | x) \quad \text{for } y_{(1)}, \dots, y_{(k)} \sim^{i.i.d.} \pi_{\theta}(\cdot | x)$$

4 Experimental Setup

In this section we detail the experimental approach used to investigate the theoretical implications of our simplified RLOO advantage function.

4.1 Baseline and Hyperparameters

As we are looking for compute advantages against the original RLOO implementation, we use the default project’s RLOO as the baseline for comparison which we will denote as ‘vanilla RLOO’ for the rest of the paper. We also use the same hyperparameters across all of our experiments to limit additional variation. Specifically, we use the important default hyperparameters of a batch size and number of gradient accumulation steps of 128; learning rate of $1e^{-3}$; a max response length of 1,024, and temperature of 1.

Because our simplified advantage function for RLOO requires our countdown task scoring function to change, we are introducing multiple points of variation against our comparison vanilla RLOO baseline. To ensure any benefits observed come from our reformulation of the simplified advantage function and zero-sum game and not from addition of length-based scoring, we perform an ablation study on each of the variations.

4.1.1 Ablation Approach and Training Details

Specifically, we compare the metrics outlined in 4.3 for each of the following combinations:

- Vanilla RLOO as implemented in milestone 2
- RLOO with the addition of a conciseness-based reward, comparing the number of tokens required to produce the correct answer with the maximum number of tokens allowed.
- RLOO with a simpler advantage formulation, setting $\hat{A}^{RLOO} = R_i$ to compare it with that derived from the zero-sum formulation
- RLOO with the conciseness-based reward and the simple advantage definition

- The Zero-Sum formulation with the reward definition in the next enumeration point

Furthermore, each of the non-vanilla RLOO combinations above are evaluated after training for 100 training steps *and* 200 training steps. Due to compute constraints, only a few runs per combination (one per number of timesteps) were able to be ran. This is certainly a limitation and scope for future work which will be touched on in 6.1 and 7.2.

4.2 Dataset Generation / Usage

In order to stay consistent with the RLOO baseline training, we opt to use the provided verifier dataset as opposed to the oracle outlined in 3.3.1; although the oracle is certainly in scope for use in follow on work.

4.3 Evaluation Metrics

We compare each of the above ablation branches and total training steps (100 and 200) on using pass@k and runtime as the primary metrics. We are using wall-clock training time as one primary comparison metric because this directly correlates to the compute efficiency of the algorithm in question. However, a more efficient algorithm may not achieve as performant of results as a more compute-heavy algorithm, so we use pass@k to evaluate the performance of the resulting modes after training for a certain amount of time. Additionally, we include the running average of the sampled reward mean at training time for each approach as this metric hints at how similar the different approaches perform or do not perform within the selected dataset.

5 Results

5.1 Quantitative Evaluation

We evaluated our Simlified Advantage RLOO appraoch on our zero-sum countdown framework, denoted as Zero-Sum for the rest of this section, against vanilla RLOO as the baseline and against the different ablation combinations using pass@k as one primary metirc. Results of these comparisons are captured in 2. These results capture a couple surprising results we should mention.

First, as can be seen when comparing 'RLOO 100 Pass@K' with 'RLOO+Length 100 Pass@K' and 'RLOO+Length 200 Pass@K' adding the length based reward to vanilla RLOO significantly degrades the performance of the resulting model. Not only does it degrade the resulting model's performance, but this degradation actually increases as the number of training steps increases. This indicates that, at least for training models starting with a SFT baseline, optimizing for response conciseness, even when only applied to correct answers, does worse than hinder learning - it makes the model's performance worse.

Additionally, as can be seen when comparing 'RLOO 100 Pass@K' with 'RLOO Simple Reward 100 Pass@K' and 'RLOO Simple Reward 200 Pass@K' using the simplified advantage expression derived from zero-sum rewards within vanilla RLOO also significantly degrades the performance of the resulting model - though not as much as adding a conciseness reward. Unlike the length-based reward effects, we see almost identical degraded performance for both the 100 and 200 step trained models, indicating that this degradation does scale with the number of training steps or that we've already reached degradation saturation at 100 training steps.

However, when we *combine* the zero-sum reward formulation with the simplified advantage calculation, we see much better results with respect to the pass@k metric. The zero-sum approach performs better than all partial-method approaches at 100 training steps, and at 200 training steps performs slightly better than RLOO at 100 steps for pass@k up to 5. At 5 however, the Zero-Sum appraoch performands drops off, as it asymptotically approaches 70%. Of note, vanilla RLOO exhibits no such aysmptotic behavior and continues to perform better on pass@k as the k values increase.

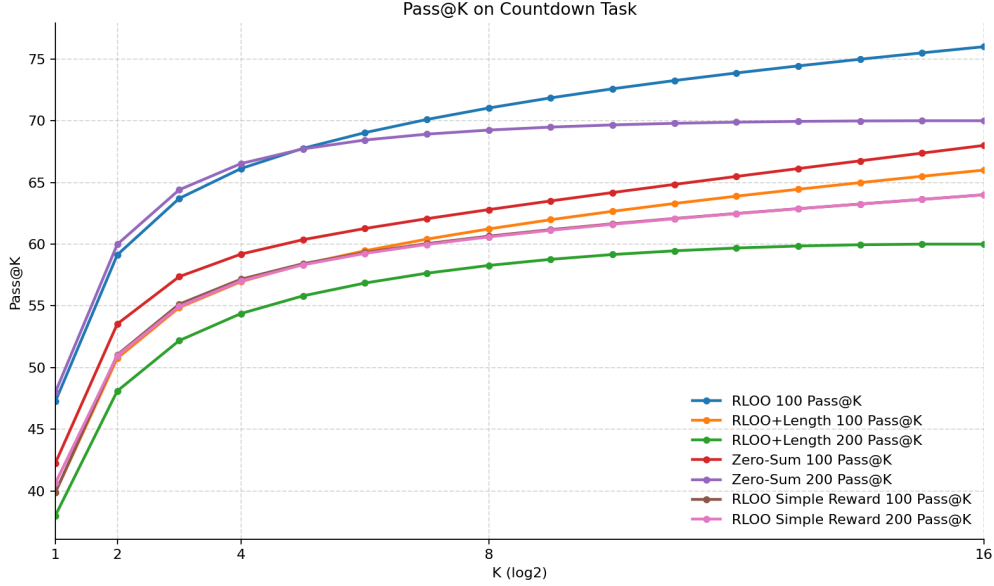


Figure 2: Pass@k Results for Ablation Study

As noted in 4.3, we are interested in the compute efficiency for models of similar performance with respect to our advantage simplification calculations. Thus, we need to map the different runs of the above models’ performance into how long it took them to run independent of the number of training steps taken.

Table 1: Compute Time Required for Full Train

Method	Time for 100 Steps	Time for 200 Steps
Vanilla RLOO	4hr 26m	8hr 52m*
Zero-Sum	3hr 36m	8hr 15m

Where the 8hr 52m for vanilla RLOO is extrapolated from the 100 time step due to compute credit constraints.

Table 1 show two key results, at least for our study on group sizes of two. The first is that the zero-sum approach is clearly more compute efficient per training step than that of vanilla RLOO. The second however, is that it takes the zero-sum approach longer to achieve the same results for pass@k for k up to around 5; and zero-sum never reaches the performance of vanilla RLOO for pass@k for k greater than 5 despite having more training steps. As expanded upon in 6.1, we believe this is due to our extremely limited sample size in terms of number of runs completed for data collection; and we were only able to investigate groups of two for zero-sum comparison. We expect both performance increases and training time decreases when the group size expands to larger numbers, and would like to continue this work measuring the amount of time it takes to reach certain pass@k values with a higher compute budget. This expectation is soundly grounded, as the Zero-Sum 200 step pass@k curve appears saturated given the large area under the curve and highly convex curve; this indicates that similar performance could have been achieved with less training steps, and thus less compute time.

5.2 Qualitative Analysis

We noticed that the length-based rewards incentivised conciseness at the cost of additional helpful information while deriving the Countdown result. That is, in vanilla RLOO, the model would output helpful ‘revelations’ about the different paths it would try:

<think>

...
2. Let's try division with division:
 $64 / 28 \sim 2.286$
This is not useful as it's not an integer.

...
6. New approach:
 $28 - 2 = 26$
 $26 + 28 = 54$
 $54 + 64 = 118$ (too large)
...
</think>
<answer>(28 + 64) - 28</answer>

We see no such 'revelations' within the RLOO variants with length based rewards:

...
First, let me try some operations:
 $58 - 5 = 53$
 $53 - 7 = 46$
 $46 - 5 = 41$
...
One more attempt:
 $58 - 5 = 53$
 $53 - 5 = 58$
 $58 - 7 = 51$
...
One last try:
 $58 - 5 = 53$
 $53 - 7 = 46$
 $46 + 5 = 51$
...

This could very well explain the degradation in performance we see as the number of training steps increases for length-based rewards in 2. Of note however, we did not see these 'revelations' within the zero-sum model either. Of course as identified earlier, zero-sum models were still able to perform relatively well after 200 training steps, and actually increased their performance as the number of training steps increased. In viewing the responses given by the zero-sum model, we see that with the simpler advantage, the length based reward actually does encourage the conciseness of responses, where it reuses previous steps in current line of thinking:

<think>
Let me try to solve this step by step:

1. First attempt:
 $79 - 66 = 13$
 $13 * 57 = 737$ (too large)

2. Second attempt:
 $66 + 57 = 123$
 $123 - 79 = 44$
 $44 + 13$ (from first attempt) = 57

I need to find a way to get 13 from the first attempt.

3. New approach:
 $79 - 66 = 13$
 $13 * 57 = 741$
 $741 - 79 = 662$

Let me try one more approach:

66 - 57 = 9
9 * 79 = 709 (too large)

Let's try:
79 - 66 = 13
13 + 57 = 70
</think>

<answer>(79 - 66) + 57</answer>

6 Discussion

6.1 Limitations

Our work does not extend to tasks that can not be represented in a zero-sum game framework, and thus promises no broad efficiency gains to the more general class of reinforcement learning algorithms without that assumption. Our work also assumed a relatively incompetent starting model across all ablations. That is, all of our results focus on the starting from the same SFT base model which is not an extremely capable model - it is still learning 'correctness'.

In addition, all of the current results are from small sample sizes due to limited compute budget. The limited compute budgets also hindered our ability to perform a true pass@k vs total runtime performance curve. Hinting at the future work section, with unlimited compute budget, we believe that the most fair comparison between the two methods would be comparing pass@k values for $k \in \{1, 16\}$ after a set amount of compute cycles (of which wall-clock time can be an approximate surrogate). That is, we believe a perfect comparison would have an $x - axis$ of compute spent on training (wall-clock time) vs pass@k performance, with different charts corresponding to different values of k .

6.2 Broader Impacts

It is important to note that although we focused on the countdown task within the scope of this paper, the only property being explored is the group advantage estimate simplification with respect to the reformulation of the task into a zero sum game. Because of this, any LLM training task that can be reformulated into a zero-sum game framework can achieve the same runtime efficiency benefits that we hint at in this paper.

Of even broader interest, we highlight here that all of the compute for our zero-sum rewards happen at data collection time. Thus, off-policy algorithms - algorithms that can reuse data collected from previous policies - stand to gain more from this zero-sum advantage simplification than on-policy algorithms do. This is because the complex portion of group scoring is done during data collection for zero-sum formulations rather than during training time, and the fraction of time spent collecting new data in off-policy algorithms is much less than that of on-policy algorithms due to the ability to reuse old data.

6.3 Difficulties

Initially, we aimed at having each response from the LLM propose both a question and an answer at the same inference time, with other responding agents attempting to beat the proposing agent. This approach was abandoned early on as it introduced additional variations that detracted from the primary contribution of this paper and the LLMs performed quite poorly at formulating proper questions, even with structured output. In lieu of this, we contribute a problem formulation oracle that randomly creates a set of integers and a target number that are guaranteed to be valid

Compute credit constraints made it difficult to perform a proper comparison across the ablation study while simultaneously providing a perfect compute-time vs pass@k performance curve. Because of this constraint, we opted to perform the ablation study to completion in order to show the benefits of the full zero-sum reward and group advantage simplification, leaving the compute-time vs pass@k curves for future work. In addition, these compute constraints limited us to using groups of only up to eight for RLOO and Zero-Sum training.

7 Conclusion

In this work we have provided a novel method for more efficient LLM post-training when group advantage estimation techniques, opening the door for smaller organizations to train LLMs that are just as performant as large organizations for less.

7.1 Finding Significance

Although results are preliminary due to computational resource constraints, the initial results strongly suggest the reformulation of LLM tasks into zero-sum games while using the simpler group advantage expressions derived in 3.2.2 can lead to significant computational efficiency while achieving the same performance. While true, there is still the possibility that the greater number of steps required to achieve this performance counteracts the computational efficiency saved within the training loop. It is because of this that we suggest multiple future directions for continued work.

7.2 Future Directions

We categorize the future work into three main categories: (1) more competent base models; (2) measuring performance gains for games that do not have a *close to zero* mean; and (3) performing a more complete study on training computation-time vs performance for groups of larger sizes.

More Competent Base Models Training on a more capable model, such as that already trained with RLOO after 100 steps, and running our study on that model could showcase even more interesting results as a more capable model would receive more dense 'conciseness' rewards.

Comparison of Zero-Sum to Large-Sum Game Performance Gains The default RLOO implementation's rewards are very close to zero for most samples. It would be interesting to measure the improvement of our zero sum implementation to a problem that does not already have a *close to 0* mean. The mean-dependent term $\frac{\mu N}{N-1}$ could be lost in the noise since this term would be relatively small for small μ .

More Complete Compute-Time Comparison As we expect Zero-Sum to exhibit faster training for larger groups, it would be worthwhile to measure the compute time required per Pass@K performance for a growing number of group sizes. Showing the compute-time vs pass@k curves while also varying the group sizes to be much larger is a promising future direction. This is also hinted at in 6.1.

8 Team Contributions

- **Joshua Slagle:** All

Changes from Proposal The only change from the originally proposed project is that the generation of the Countdown problem formulation by the LLMs themselves proved to be very inefficient, so we ended up moving this capability outside of the model's purview and into a separate piece of code.

References

- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. arXiv:2402.14740 [cs.LG] <https://arxiv.org/abs/2402.14740>
- Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Zheng Yuan, Yong Dai, Lei Han, Nan Du, and Xiaolong Li. 2025. Self-playing Adversarial Language Game Enhances LLM Reasoning. arXiv:2404.10642 [cs.CL] <https://arxiv.org/abs/2404.10642>
- Kefan Dong and Tengyu Ma. 2025. STP: Self-play LLM Theorem Provers with Iterative Conjecturing and Proving. In *Forty-second International Conference on Machine Learning*. <https://openreview.net/forum?id=zWArMedNuW>

- Hao Ma, Tianyi Hu, Zhiqiang Pu, Boyin Liu, Xiaolin Ai, Yanyan Liang, and Min Chen. 2024. Coevolving with the Other You: Fine-Tuning LLM with Sequential Cooperative Multi-Agent Reinforcement Learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=0o0CoZFVK3>
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290 [cs.LG] <https://arxiv.org/abs/2305.18290>
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] <https://arxiv.org/abs/1707.06347>
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv abs/2402.03300* (2024). <https://api.semanticscholar.org/CorpusID:267412607>
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.
- Vighnesh Subramaniam, Yilun Du, Josh Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent Finetuning: Self Improvement with Diverse Reasoning Chains. *ArXiv abs/2501.05707* (2025). <https://api.semanticscholar.org/CorpusID:275458404>
- Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* 575, 7782 (2019), 350–354.
- Tong Zhu, Baiting Chen, Jin J Zhou, Hua Zhou, Sriram Sankararaman, and Xiaowu Dai. 2026. Competitive Multi-Agent Delegation for LLM Reasoning. <https://openreview.net/forum?id=nDdpp0285M>