Extended Abstract

Motivation Large language models (LLMs) face critical challenges in aligning their outputs with nuanced human preferences and effectively solving structured tasks requiring explicit logical reasoning. Traditional supervised approaches, while effective in initial stages, often fall short in fine-tuning models for optimal alignment and reasoning precision. Reinforcement learning (RL), especially preference-based optimization techniques, presents a promising avenue to bridge this gap by leveraging explicit human feedback and structured rewards. Motivated by these potential benefits, our project explores advanced reinforcement learning strategies to fine-tune LLMs, specifically addressing tasks of instruction following and mathematical reasoning.

Method We explore preference-based language model alignment for better **Insruction Following** using two strategies: Direct Preference Optimization (DPO) and a multi-objective extension to the RLOO framework. DPO treats preference learning as a classification task using pairwise human feedback, while RLOO learns a Bradley-Terry reward model from preferences and combines it with auxiliary objectives. These objectives are encoded via prompt-based scoring and rule-based constraints, and combined using scalarization and Pareto reasoning. For the **Countdown** task, we explored two RL algorithms: RLOO (REINFORCE Leave-One-Out), which provides an unbiased policy gradient estimate with a leave-one-out baseline, and a hybrid extension combining PPO (Proximal Policy Optimization) with RLOO's baseline calculation.

Implementation Our implementations are based on the Qwen2.5-0.5B Base model, developed by Alibaba Cloud. For **Instruction Following**, we fine-tune a Qwen-2.5-0.5B model using SFT and apply DPO with 100 and 500 samples from UltraFeedback. For the extension, we apply RLOO optimization on SFT and incorporate multi-objective rewards. All models are evaluated on 100 UltraFeedback samples using the Nemotron 70B reward model. For the **Countdown** Task, we used the reward function proposed by Gandhi et al. (2025b), which combines both format and verification scores to provide structured reward signals. To facilitate training with larger sample sizes and improve stability, we employed several memory-optimization strategies.

Results On **Instruction Following**, DPO-100 achieves the highest win rate, with SFT performing comparably and DPO-500 underperforming. In the multi-objective setup, combining fluency and consistency rewards consistently improves win rate, while grammar offers moderate gains. Repeated Patterns and Chaotic Sequences objectives yield lower or negative impacts. Correlation analysis shows positive alignment among fluency, consistency, and grammar, and negative correlations between stylistic constraints and core language quality. On the **Countdown** task with 1000 eval prompts, the SFT model achieved the highest rewards (0.3330), while RLOO maintained a comparable high rewards (0.3168) with fewer null outputs than PPO, which struggled with a high rate of empty equations or wrong output with a 0.3132 rewards.

Discussion Moderate preference tuning improves alignment in **Instruction Following**, but excessive supervision (as in DPO-500) may harm both reward scores and win rates. Multi-objective optimization enhances performance when focused on high-level language quality, but rigid penalties can conflict with human preferences. Fine-tuning large language models on the **Countdown** task using RLOO and PPO is challenging given the high variance in the leave-one-out (LOO) baseline, even with PPO's clipped objective. Evaluation results suggest that SFT offers a strong structural baseline, and while RLOO retains robustness in output formatting, PPO suffers from stability issues that reduce its effectiveness in reasoning tasks.

Conclusion Our findings show that preference-based alignment benefits from carefully calibrated supervision and balanced reward design in **Instruction Following**. Combining Bradley-Terry models with targeted auxiliary objectives offers a flexible and interpretable approach to aligning language models with nuanced user expectations. Supervised fine-tuning on the **Countdown** task highlighted the importance of early stopping to prevent overfitting and boost generalization. When applying online RL methods like RLOO and PPO, we found that managing GPU efficiency is critical for scaling roll-out sizes and improving training stability. Overall, supervised fine-tuning remains the most reliable approach for the Countdown task, while RL methods like RLOO/PPO show promise but require further optimization to improve arithmetic correctness without sacrificing response consistency.

Default Project: RL Fine-Tuning of Language Models

June Zheng
Institute for Computational and Mathematical Engineering
Stanford University
yujunz@stanford.edu

Yihan Zhao
Department of Biomedical Data Science
Stanford University
yhanzhao@stanford.edu

Zixin Li
Department of Electrical Engineering
Stanford University
zixinl4@stanford.edu

Abstract

Large language models (LLMs) have achieved remarkable capabilities across a wide range of natural language tasks. However, aligning their outputs with human preferences remains a fundamental challenge—especially when desirable responses are task specific or structured. In this project, we explored whether reinforcement-learning (RL) fine-tuning can close this gap on two demanding benchmarks—open-ended instruction following and the numeric Countdown game—using the 0.5-B-parameter Qwen-2.5 base model. Starting with supervised fine-tuning (SFT) on 470 k SmolTalk examples, we applied Direct Preference Optimization (DPO) and a Bradley-Terry REINFORCE-Leave-One-Out (RLOO) learner augmented with multi-objective rewards (fluency, consistency, grammar, style). With just 100 preference pairs, DPO raised the Nemotron-70B win-rate to 83 % and score on leaderboard reaches the 2nd place. On Countdown, SFT over 1 k curated solutions achieved a verification-weighted reward of 0.333 on 1 000 held-out prompts, RLOO narrowed the gap to 0.317 while maintaining structured output, whereas a PPO-RLOO hybrid saw high-variance updates and more malformed equations.

1 Introduction

In this project, we investigate the effectiveness of reinforcement learning (RL) fine-tuning methods in enhancing the performance of the Qwen2.5-0.5B language model, specifically focusing on instruction following and mathematical reasoning tasks. We initially employ Supervised Fine-Tuning (SFT) as a foundational step, enabling the model to acquire structured task-specific knowledge and behavioral norms from curated datasets. Subsequently, we apply advanced RL techniques, including Direct Preference Optimization (DPO) and REINFORCE Leave-One-Out (RLOO), to further align model outputs with human preferences and structured task requirements. Our research introduces novel extensions to the standard RL framework, integrating auxiliary reward signals such as fluency, grammatical correctness, and logical consistency through scalarization and Pareto optimization methods. Moreover, we adopted a hybrid version of RLOO and PPO when enhancing model's math reasoning abilities.

For the instruction-following task, we utilize datasets such as Smoltalk and UltraFeedback to train and evaluate the model's alignment with human preferences. In the mathematical reasoning task (Countdown), we address the model's ability to perform complex arithmetic and logical operations using datasets specifically designed to test structured reasoning skills. Through systematic experimentation, including ablation studies and performance analysis across various reward configurations, we aim to identify optimal strategies for aligning language models with nuanced human preferences and structured reasoning capabilities. Our findings provide insights into effective RL fine-tuning methodologies, highlighting critical considerations for improving language model performance across diverse and complex tasks. On the leader board, our score for instruction following task reaches **2nd place**.

2 Related Work

Instruction Following:

Our work on instruction following builds on foundational approaches in aligning language models with human preferences. Supervised Fine-Tuning (SFT)Ouyang et al. (2022) has been widely adopted for initial alignment using expert demonstrations. Direct Preference Optimization (DPO)Rafailov et al. (2023) introduces an efficient alternative that directly models pairwise preferences without relying on an explicit reward model, enabling stable and scalable fine-tuning. Bradley-TerryBradley and Terry (1952) models have also been leveraged to learn latent reward functions from binary preferences, serving as the basis for ranking-based methods like RLOO.

While most preference-based alignment methods focus on optimizing a single scalar reward, recent research in multi-objective reinforcement learning (MORL)Roijers et al. (2013) has emphasized the importance of balancing competing objectives such as fluency, consistency, factuality, and safety. Methods such as scalarization, constraint-based optimization, and Pareto front learning have been proposed to handle these trade-offsVan Moffaert and Nowé (2014). However, their application in language alignment remains limitedYang et al. (2024). Our work addresses this gap by integrating auxiliary reward signals with Bradley-Terry-based preference models, enabling a more interpretable and flexible alignment pipeline.

Mathematical reasoning:

Mathematical reasoning is a key challenge in evaluating large language models (LLMs). Early work, such as GPT-3 by Brown et al. (2020), showed that few-shot prompting enables basic arithmetic capabilities, but more structured reasoning requires stronger supervision and feedback mechanisms.

Recent efforts like the Countdown task from Gandhi et al. (2025b) introduced verifier-based reward functions to evaluate both correctness and format, providing a structured benchmark for multi-step problem-solving. Supervised fine-tuning (SFT) on such data has proven effective for initialization.

To further enhance performance, reinforcement learning (RL) approaches have been explored. Ahmadian et al. (2024) introduced REINFORCE Leave-One-Out (RLOO), offering unbiased gradients with reduced variance, while Schulman et al. (2017) proposed Proximal Policy Optimization (PPO) for stable updates. Combining these—e.g., PPO with RLOO's baseline—has shown promise in training stability and generalization Team (2024).

3 Method

3.1 Instruction Following:

Basic Implementation:

The initial stage of any reinforcement learning (RL) pipeline in language tasks involves Supervised Fine-Tuning (SFT). This stage uses the same next-token prediction objective as in the pre-training phase, but notably, the loss is not computed over the query tokens. Instead, the supervised learning objective is optimized over query–completion pairs *(x,y)* sampled from an expert distribution. When applied to preference datasets, the completion y typically corresponds to the preferred response y_w, also known as Pref-FT. This objective can be formalized as follows:

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

For further optimization, we implement th Direct Preference Optimization(DPO) objectivesRafailov et al. (2023) show that, by using a smart reward reparameterization, a constrained reinforcement learning (RL) objective can be reformulated as a supervised preference classification task using human preference data. Specifically, the DPO (Direct Preference Optimization) loss is defined as:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

where x is the input prompt, y_w is the preferred response, y_l is the less preferred response, π_θ denotes the learnable policy, and π_{ref} is the reference policy.

Extension Implementation:

Traditional preference-based reinforcement learning (RL) typically aims to optimize a single scalar reward that reflects overall user preference. However, many real-world language tasks involve trade-offs between multiple desirable attributes.

The basic RLOO implementation on preference dataset is optimized on Bradley-Terry reward, which is learned from the preferences between preferred and unprepared samples. More formally, the objective can be written as follows:

$$\mathcal{L}_{\text{BT}}(\phi) = \max_{\phi}\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{pref}}}\left[\log\sigma\left(r_\phi(\mathbf{x},\mathbf{y}_w) - r_\phi(\mathbf{x},\mathbf{y}_l)\right)\right]$$

The multi-objectives are applied to optimize RLOO algorithm to compare with each other. We apply various objectives that reward and penalize language patterns and behavior (fluency, language consistency or grammar) using existing reward function or hard-coded utility function. These objectives are combined with Bradley-Terry preference model through techniques like weighted summation. The objectives we're expanding in extension section are:

1. **Grammar, Consistency and Fluency**: in-context learning, prompt a Qwen-2.5-0.5B model to score the response on Grammar from the scale of -5 to 0.
2. **Repeat Patterns**: Rule-based, we match long sequences in response, if there're repeated pattern/sequences that are longer than 30 tokens, the reward on this part is minused by one.
3. **Chaotic Sequences**: Rule-based, if our hard-coded detector detects chaotic sequences in the end of the generated response, the overall reward minus 5, otherwise kept the same.

3.2 Math Reasoning

For enhancing the math reasoning abilities of LLM, we focused on improving the model's ability to perform multi-step mathematical problem-solving with the task of Countdown. In Countdown, the model is given a set of input numbers and a target number, and is prompted to logically determine a sequence of arithmetic operations that transform the input numbers into the target number. This task challenges and evaluates the LLM's ability to plan, decompose, and execute a series of mathematical steps in order to reach the correct answer.

We are going to train the LLM in two different stages. The first stage uses supervised learning to warm-start the language model. This helps to stabilize training for further reinforcement learning fine-tuning. It also allows the model to learn structured task formats and instructions. The second stage is the use of reinforcement learning algorithm to further fine-tune the model and improve its generalization abilities.

For the math reasoning task, we utilized two reinforcement learning techniques: **RLOO** (REINFORCE Leave-One-Out) Ahmadian et al. (2024) and a variation of **PPO** (Proximal Policy Optimization) for the extension Schulman et al. (2017).

RLOO is a policy gradient estimator based on REINFORCE that introduces a baseline to reduce variance. This baseline is the average reward of all other samples drawn from the same policy, hence making it a leave-one-out baseline. It uses this baseline to derive the reward advantage for each sampled roll-out during the training iterations.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})\right)$$

Figure 1: Vanilla REINFORCE policy gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\underbrace{\sum_{i=1}^{N}}_{\text{samples from policy}}\sum_{t=1}^{T}\nabla_\theta \underbrace{\log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}_{\text{policy log likelihood}}\left(\left(\underbrace{\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})}_{\text{reward to go}}\right)\underbrace{-\,b}_{\text{baseline}}\right)$$

Figure 2: REINFORCE policy gradient with baseline

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

Figure 3: RLOO policy gradient

The RLOO algorithm comes with both strengths and limitations. On the positive side, its leave-one-out baseline effectively reduces variance compared to vanilla REINFORCE. Additionally, because the baseline is constructed independently of each sampled roll-out, RLOO remains an unbiased estimator of the true policy gradient. However, RLOO still suffers from relatively high variance, especially when the number of roll-out samples is relatively small, given that the quality of the baseline depends heavily on the sample size. Another limitation is that RLOO lacks built-in KL regularization, meaning it does not inherently prevent the policy from diverging too far from the supervised fine-tuning (SFT) model.

As an **extension**, we decided to use a hybrid approach that integrates the **PPO** proximal policy optimization objective with the leave-one-out baseline from RLOO. The motivation for this combination is to leverage PPO's inherent KL-regularization that constrains the updated policy from deviating too much from the reference model, while simultaneously benefiting from the reduced variance and unbiased properties of RLOO's gradient estimation. We hypothesized that this approach enables more stable and efficient policy updates. Therefore, in the objective 4, $\hat{A}^{\pi_\theta}$ is estimated using the leave-one-out baseline from RLOO.

$$\tilde{J}(\theta') \approx \sum_{t,i} \min\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}), \text{clip}\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})\right)$$

Figure 4: Proximal Policy Optimization Objective

4 Experimental Setup

4.1 Qwen2.5 Base Model

Our experiments are based on the Qwen2.5-0.5B Base model, Team (2024), developed by Alibaba Cloud. It is the latest series of Qwen large language models.

4.2 Instruction Following:

For basic implementation, we fine-tune the Qwen 2.5 0.5B base model on the Smoltalk dataset (470k samples) using Supervised Fine-Tuning (SFT), and optimize the SFT model with DPO optimization using 100/500/1000 samples from Ultrafeedback dataset. The models are than evaluated on 100 samples from the Ultrafeedback binarized dataset.

For extension implementation, the pipeline is trained on 100/500/1000 samples from Ultrafeedback, and is evaluated on 100 samples from Ultrafeedback binarized dataset.

For both basic and extension implementation, our model is benchmarked against the Qwen-2.5-0.5B-instruct using the Llama 3.1 Nemotron 70B Reward Model.

4.3 Math Reasoning

For the **supervised fine-tuning stage**, we warm-started the Qwen2.5 base model with the cog_behav_all_strategies (https://huggingface.co/datasets/Asap7772/cog_behav_all_strategies) dataset from HuggingFace. This dataset contains 1,000 training data and 200 test data. Each training data example contains a query text that describes the countdown task and a completion text that logically solves the problem 12.

For both the **RLOO and PPO extension** fine-tuning stage, we used the TinyZero dataset https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4 from HuggingFace. This dataset comprises of approximately 490,000 Countdown prompts, each consisting of a list of numbers and a target value to be computed. To ensure consistency in input representation and to promote learning stability, we adapted the same query formatting used in the cog_behav_all_strategies dataset.

For the Countdown task, we leveraged the rule-based reward function provided by Gandhi et al. (2025a). The reward function gives 0.1 for the correct format of the answer and 1 for getting the correct solution. The format score encourages the LLM to provide a parseable answer.

4.4 Evaluation Metric:

For instruction following, the evaluation framework employs the NVIDIA Nemotron 70B reward model to provide comprehensive performance assessment through multiple complementary metrics:

1. **Win Rate**: Percentage of cases where the trained model's responses are preferred over a reference model (Qwen2.5-0.5B-Instruct), providing a direct measure of relative performance improvement.
2. **Average Model Reward**: Mean reward score for the trained model's responses, indicating absolute quality as measured by the sophisticated Nemotron reward model.
3. **Average Reference Reward**: Baseline performance metric from the reference model for comparative analysis.

For the Countdown task, we used a weighted average of the format score and the verification score to evaluate our model's performance.

Memory Efficiency Setup:

Given that we used an online learning approach with both RLOO/PPO, which involves collecting new experiences during each training iteration, it is essential to manage computational and memory efficiently. Online RL methods often require larger batch/roll-out sizes to ensure stable gradient estimates. However, such larger sample size can quickly exceed GPU memory limits, especially when using large language models. To address this, we utilized a combination of memory-efficient training techniques for efficient memory management to enhance learning stability.

1) Mixed precision training: leverages lower-precision formats (EX: FP16) to reduce memory footprint without significant loss in numerical stability.

2) Gradient Accumulation: Accumulate gradients over multiple batches before applying an optimizer step.

3) CPU Offloading with FSDP (Fully Sharded Data Parallel): automatically moves model parameters and gradients between GPU and CPU to help to release memory in GPU when not being actively used.

5 Results

5.1 Instruction Following:

Model Evaluation:

For the basic implementation, we show win rate, average model reward, average reference rewards for SFT, DPO(trained on 100 and 500 samples). All win rate of three models are shown in Figure 5, average model/reference reward are shown in the same picture for each model in Figure 6:

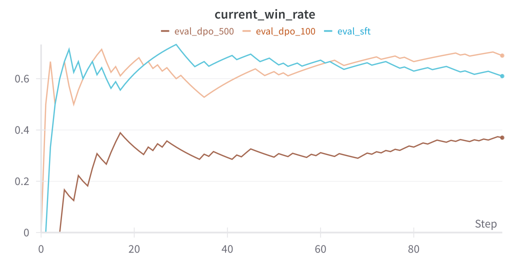
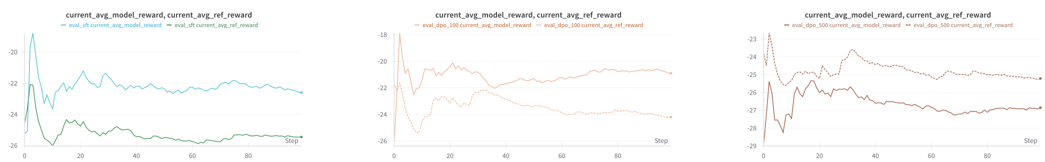


Figure 5: Instruction Following: Win Rate for all models



(a) SFT (b) DPO-100 (c) DPO-500

Figure 6: Instruction Following: Average model/reference reward for each model

For extension implementation, we first train an RLOO model with Bradley-Terry Rewards as our baseline, Then we applied some simple hard-coded reward functions and prompt-based llm judgement as our multi-objectives. In Table 1, we could see how applying different objectives could impact our model's win rate over reference model.

Model	Grammar	Consistency	Fluency	Repeated Patterns	Chaotic Sequences
RLOO-100	0.785	0.818	0.830	0.770	0.735
RLOO-500	0.778	0.844	0.832	0.731	0.751
RLOO-1000	0.775	0.813	0.824	0.702	0.706

Table 1: Win-rates of models trained with various objectives over reference model

We also conduct Pareto-optimality analysis on different objectives here by evaluating not only the win rate and Bradley-Terry rewards but also all other objectives. By bootstraping series of model, we apply regression on all pairs of changing magnitude between objectives and obtain their correlation coefficient, the results could be found in heatmap 7.

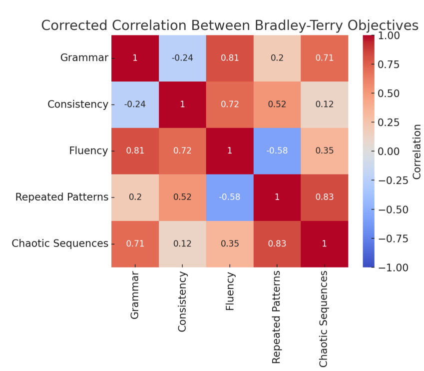


Figure 7: Instruction Following Extension: Correlation of Multi-objectives

Analysis For Basic Implementation, Figure 5 shows that DPO-100 achieves the highest overall win rate, closely followed by SFT, while DPO-500 consistently performs below. This indicates that moderate preference-based fine-tuning (with 100 samples) can slightly enhance alignment over SFT, but excessive preference supervision (500 samples) may lead to degraded performance—possibly due to overfitting or misaligned reward learning.

Result Table 1 could be visualized as the heatmap 8 below:

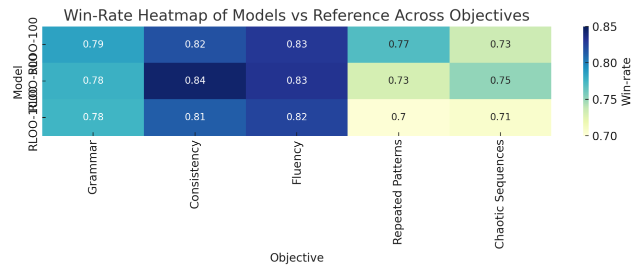


Figure 8: Instruction Following Extension: Win Rate when applying multi objectives

The heatmap shows that Fluency and Consistency are the most effective objectives, consistently yielding the highest win-rates across all RLOO variants. These signals closely align with human preferences, improving overall response quality. Grammar offers moderate gains, suggesting that users tolerate minor errors if fluency is preserved.

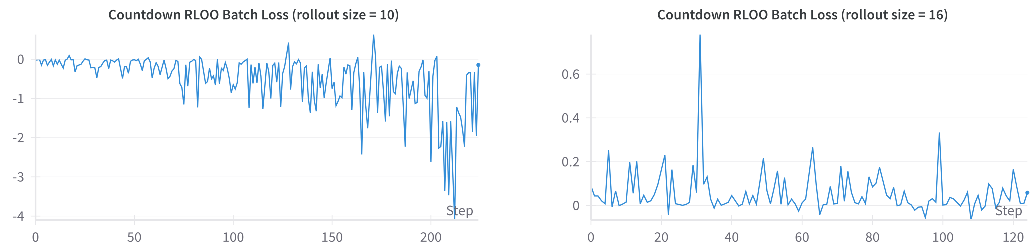
In contrast, Repeated Patterns and Chaotic Sequences show lower and more variable performance. Over-penalizing these may suppress useful stylistic elements or model flexibility. Notably, increasing training scale (from RLOO-100 to RLOO-1000) doesn't always improve results—some objectives peak earlier, indicating diminishing returns. Overall, fluency and coherence are key drivers of alignment, while secondary signals require careful tuning.

5.2 Math Reasoning

Model Training:

During the **supervised fine-tuning** stage, we used a batch size of 5, learning rate of 1e-05, and a maximum tokenizer encoding length of 1024. We initially trained the model for 50 epochs and observed a notable trend of diminishing returns beyond epoch 20. This plateau suggests that further training might risk overfitting to the SFT dataset. As a result, we selected the checkpoint at epoch 20 to serve as the initialization model for subsequent reinforcement learning fine-tuning.

When further fine-tuning the SFT model with the **RLOO** algorithm, we used a batch size of 1, learning rate of 1e-05, gradient accumulation batch count of 2, and roll-out sample size of 16. The initial setup was a sample size of 10 and summing over all the sequence log probabilities in the objective calculation. However, we noticed a high variance in training loss 9a. Therefore, we changed the roll-out sample size to 16 for the final implementation. The scale-up in sample size was made feasible through the use of memory optimization techniques, including mixed-precision training, gradient accumulation, and tensor offloading with Fully Sharded Data Parallel (FSDP). We can observe in 9b that by scaling up the sample size, the training loss is much more stable. Moreover, we also changed to averaging over all the sequence log probabilities instead of summing to avoid penalizing a response due to its sequence length.



(a) Rollout Size = 10 (b) Rollout Size = 16

Figure 9: Countdown RLOO Training Loss

For fine-tuning the SFT model using the hybrid **PPO** algorithm augmented with RLOO's reward calculation, we used a double-network framework. In this framework, the network used to collect roll-out experience lags behind the current policy by 5 update steps. Moreover, we used a batch size of 1, learning rate of 1e-05, roll-out sample size of 5, and gradient accumulation batch count of 50. Even with using PPO to limit the policy from deviating too much at each gradient step, we observed that it still exhibits substantial variance in training loss. We think PPO would also require a large roll-out sample size to stabilize the learning process, similar to RLOO. Due to the high computational demands of online Reinforcement Learning methods and our limited compute resources, we chose prioritize fine-tuning Countdown with RLOO and the extension for the Instruction-Following task.

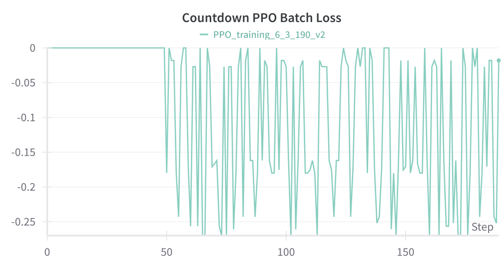


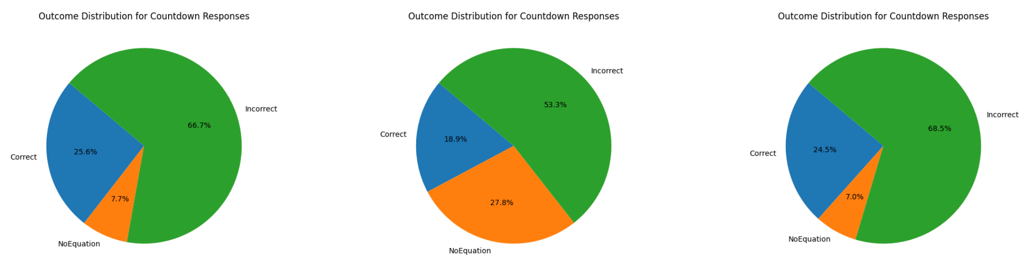
Figure 10: Countdown PPO Training Loss

Model Evaluation:

The evaluation results from our experiments on the Countdown task reveal key insights into the effectiveness of different fine-tuning strategies. As shown in the table 2, the supervised fine-tuned model (SFT) achieves the highest reward scores on both the 1000 new prompts (0.3330) and 200 old prompts (0.5270), demonstrating strong baseline performance and generalization. RLOO-based models follow closely, achieving consistent scores across datasets, which highlights the benefits of variance-reducing techniques in arithmetic reasoning tasks. PPO-based models show competitive performance on the old 200 prompts but suffer from a performance drop on the new 1000 prompts, suggesting possible overfitting or instability during training. Notably, all models perform better on the old 200 prompts, indicating a generalization gap that may stem from distribution shifts or prompt structure differences. These results suggest that while supervised fine-tuning provides a robust foundation, reinforcement learning methods—particularly those with unbiased or stabilized objectives—offer valuable improvements. However, further tuning and hybrid strategies may be needed to bridge the gap in performance between known and novel inputs.

Model	**200 Prompts**	**1000 Prompts**
SFT	**0.5270**	**0.3330**
RLOO	0.4610	0.3168
PPO	0.4850	0.3132

Table 2: Evaluation results for Countdown task



(a) SFT (b) PPO (c) RLOO

Figure 11: Outcome distribution of Countdown task responses for different training strategies.

We visualize the outcome distributions for three models—SFT, PPO, and RLOO—on the Countdown task11. The SFT model achieves the highest correctness rate (25.6%) while maintaining a low no equation output rate (7.7%), indicating its reliability in generating structured and interpretable responses followed by the instructions. In contrast, the PPO model exhibits a increase in no equation error (27.8%). RLOO demonstrates a more balanced behavior, which retains a low no equation rate (7.0%) and maintains a correctness rate (24.5%) close to that of SFT. These observations highlight that while PPO may struggle with stability in arithmetic reasoning under limited compute, RLOO offers a more consistent and structure-preserving alternative for reinforcement learning fine-tuning when properly configured. Overall, these trends underscore the importance of reward design and rollout stability when applying RL to structured reasoning tasks.

6 Discussion

6.1 Instruction Following

In the basic setup, DPO=100 achieves the best performance, with SFT slightly behind and DPO-500 performing the worst—suggesting that more preference data does not guarantee better alignment and may lead to overfitting.

In the extension, combining Fluency and Consistency objectives consistently improves win rates. Grammar adds moderate value, while Repeated Patterns and Chaotic Sequences often reduce performance. Correlation analysis shows these two negatively impact fluency and consistency, highlighting trade-offs in multi-objective reward design.

6.2 Math Reasoning:

For the Countdown task, one of the primary challenges encountered during fine-tuning with RLOO or PPO lies in the instability of the training loss, which significantly affects convergence and overall learning performance. This instability largely arises from the high variance inherent in the leave-one-out baseline estimation. This high variance makes the learning signal very noisy, thereby degrading training stability. Although PPO introduces a clipped objective to limit how drastically the policy is allowed to change during each update, it does not resolve the variance issue when combined with the leave-one-out baseline calculation.

Hence, GPU constraints pose a significant limitation in the context of online reinforcement learning when fine-tuning the SFT model using RLOO or PPO. Online RL requires the policy to continuously interact with the environment to generate new experiences. These requirements inflate memory consumption and GPU utilizations. These demands then get exacerbated when working with large language models, which typically involve long context lengths, causing further challenges.

The evaluation results from our experiments on the Countdown task reveal important insights into model performance. The supervised fine-tuned model (SFT) consistently demonstrated the highest performance across both new and old prompts, suggesting strong baseline capabilities. Models using the REINFORCE Leave-One-Out (RLOO) strategy achieved moderately good results, showcasing the effectiveness of unbiased policy gradient estimates in improving arithmetic reasoning. The differences in scores between old and new prompts across all models highlight the generalization challenge in math reasoining tasks. Future work should focus on addressing generalization gaps and further refining reinforcement learning approaches to achieve improved robustness and consistency in mathematical reasoning tasks.

7 Conclusion:

7.1 Instruction Following:

Our experiments demonstrate that preference-based fine-tuning can benefit from moderate supervision but may degrade when overapplied. In multi-objective extensions, aligning models with high-level language qualities like fluency and consistency provides the most reliable gains. However, incorporating rigid pattern constraints requires careful calibration to avoid unintended side effects. These findings highlight the importance of both reward design and objective balancing in al

7.2 Math Reasoning:

During supervised fine-tuning of the Countdown task, we found that the use of effective early stopping criteria plays a crucial role in preventing overfitting and enhancing the model's generalization abilities. This was reflected in the strong performance exhibited by our SFT model during evaluations. Moreover, the key takeaway from fine-tuning Countdown with online reinforcement learning algorithms like RLOO and PPO, it is important to effective manage GPU efficiency in order to scale up batch/roll-out size to improve learning stability, which is demonstrated in 9b. Due to computational resource constraints, we were unable to train the RLOO algorithm for an extended number of iterations. As a result, further training and more extensive hyperparameter tuning of RLOO are deferred to future work to fully explore its convergence behavior on the Countdown task.

When evaluating on the held-out test prompts, the supervised fine-tuned model remains the strongest baseline, achieving the highest performance on both evaluation sets. RLOO-based models provide a promising middle ground, while PPO-based approaches suffer from stability issues under compute constraints. The overall trend reveals that models struggle to generalize well from old to new prompts, emphasizing the importance of training diversity and evaluation robustness. Future efforts should prioritize techniques that enhance generalization, such as curriculum learning and prompt augmentation.

8 Team Contributions

- **June Zheng:** Countdown Dataset Implementation; Countdown SFT Implementation & Training; Countdown RLOO Implementation & Training; Countdown PPO Extension Implementation & Training; Paper Writing & Analysis For the Countdown Task
- **Yihan Zhao:** Methodology design, Codes implementation, Experiment and Manuscript drafting for instruction following section.
- **Zixin Li:** Countdown evaluation implementation and analysis, DPO code implementation, Experiment set up, poster design, and paper writing.

Changes from Proposal Changes were made to isolate the implementation based on the task for easier debugging.

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`

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

Tom Brown et al. 2020. Language Models are Few-Shot Learners. *Advances in neural information processing systems* (2020).

Kanishk Gandhi et al. 2025b. Cognitive-Behaviors: Structured Reward for Multi-step Reasoning in LLMs. *arXiv:2501.12345* (2025).

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025a. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. arXiv:2503.01307 [cs.CL] `https://arxiv.org/abs/2503.01307`

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2023), 53728–53741.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

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`

Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. `https://qwenlm.github.io/blog/qwen2.5/`

Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3512.

Kailai Yang, Zhiwei Liu, Qianqian Xie, Jimin Huang, Tianlin Zhang, and Sophia Ananiadou. 2024. Metaaligner: Towards generalizable multi-objective alignment of language models. *arXiv preprint arXiv:2403.17141* (2024).

A Sample Example from Cog_behav_all_Strategies Dataset

```
A conversation between User and Assistant.
The user asks a question, and the Assistant
solves it. The assistant first thinks about
the reasoning process in the mind and then
provides the user with the answer.
User: Using the numbers [95, 36, 32],
create an equation that equals 91. You can
use basic arithmetic operations (+, -, *,
/) and each number can only be used once.
Show your work in <think> </think> tags.
And return the final answer in <answer>
</answer> tags, for example <answer> (1 +
2) / 3 </answer>.
Assistant: Let me solve this step by step.
```

```
<think>
Let me analyze this step by step:

1. First, let's try to get close to 91 by
working with the larger numbers:
95 is closest to 91, only 4 away. Maybe we
can subtract something.

2. Looking at 36 and 32:
36 - 32 = 4
This could be useful since we need to reduce
95 by 4!

3. Let's verify:
If we do 95 - (36 - 32) = 95 - 4 = 91
This works perfectly!

4. Let's verify the solution working forward:
First: 36 - 32 = 4
Then: 95 - 4 = 91

This gives us our target of 91.
</think>

<answer>95 - (36 - 32)</answer>
```

Figure 12: Example Training Example from Cog_behav_all_Strategies

B SFT Training Loss for Countdown Task

Figure 13: Countdown SFT Training Loss