Extended Abstract

Motivation Large language models (LLMs) trained using supervised fine-tuning (SFT) often struggle with multi-step arithmetic reasoning tasks that require both syntactical correctness and exact precision of mathematical operations. Reinforcement learning algorithms, such as Reinforce Leave-One-Out (RLOO) Ahmadian et al. (2024), can offer a powerful mechanism to optimize policies directly on execution outcomes. However, standard reinforcement learning environments often utilize sparse, binary reward functions (0.0 for failure, 1.0 for success) that present an extremely challenging exploration task. In tasks like the Countdown game, where one must combine a set of integers using basic arithmetic operations to reach a specified target value, sparse rewards can lead to very inefficient policy gradient updates, slow convergence, and high sample variance. This project seeks to address the sample inefficiency of sparse post-training RL by creating a continuous, denser reward signal to better facilitate policy updates.

Method We introduce a dense reward-shaping framework, the *Bounded Proximity Reward*. Instead of evaluating equations on a purely binary outcome, our reward function injects a smooth, continuous proximity bonus into non-matching but syntactically valid operations. When a model generates a mathematically valid formula that fails to reach the target integer, it receives an incentive score that is inversely proportional to the absolute difference between the calculated outcome and the true target value. Most importantly, this score is strictly bounded below the value of a perfect match to eliminate optimization vulnerabilities. For example, earlier versions of this reward shaping framework ran into issues such as subgoal farming, which is where an agent might intentionally generate short, incorrect paths to maximize the intermediate reward rather than solving the global problem.

Implementation Our architecture builds upon a baseline post-training pipeline using the default project's SFT model configuration using the $Qwen2.5 - 0.5B$ model Team (2024), optimized over the Countdown dataset. The reinforcement learning phase is orchestrated via the default RLOO framework using vLLM Kwon et al. (2023). We compare two policy setups: (1) a base RLOO implementation relying on standard sparse feedback, and (2) our dense RLOO implementation utilizing the bounded proximity reward. Training runs are carried out using standard hyperparameter constraints, utilizing a constant learning rate of 1×10^{-5}, an effective batch size of 128, and a KL divergence penalty Stiennon et al. (2020).

Results Our dense incentive structure yields a significant optimization acceleration. Quantitative evaluation using multi-sample inference shows that the Dense RLOO model achieves a higher $pass@1$ accuracy of **50.38%**, a significant increase over the base RLOO model's $pass@1$ accuracy of **41.22%**. However, multi-sample $pass@k$ scaling analysis reveals that for a k value of 16, the performance of the Base RLOO model scales to **63.60%**, whereas our Dense RLOO model's scaling curve flattens more quickly, culminating in a $pass@16$ score of **62.00%**.

Discussion An analysis of the training dynamics confirms that this difference in scaling was caused by mode collapse. Under a sparse reward paradigm, the base policy is able to maintain high token entropy due to wider exploration, creating a very diverse set of generation samples across large values of k. Conversely, the explicit guidance provided by the dense proximity reward quickly forces the policy to traverse more predictable high-reward paths. While this elevated the first-pass accuracy, it leads to near-identical generation duplication across multiple inference iterations, seen in some samples logs in the appendices. The training logs reveal that this collapse occurs near training step 50, characterized by a sharp drop in the mean importance weight.

Conclusion This study demonstrates that dense reward shaping provides an effective mechanism for boosting first-turn language model reasoning capabilities, making it valuable for deployments where single-sample inference costs may dominate. To achieve higher $pass@k$ without sacrificing these $pass@1$ gains, future work will explore entropy maximization or multi-sample uniqueness penalties inside the policy update.

Bounded Proximity Rewards: Accelerating Post-Training Reinforcement Learning for Arithmetic Reasoning without Policy Divergence

Ian Chen
Department of Computer Science
Stanford University
ianychen@stanford.edu

Abstract

Post-training reinforcement learning represents an important area in aligning large language models for complex mathematical and reasoning tasks. While sparse rewards offer clean, simple optimization objectives, they can suffer from exploration inefficiencies in math environments where a single incorrect token can invalidate an entire trajectory. In this paper, we explore reward shaping within the Reinforce Leave-One-Out (RLOO) framework applied to the Countdown task. We present a bounded proximity reward function that accelerates training convergence and pushes single sample accuracy from 41.22% to 50.38% using a $Qwen2.5 - 0.5B$ backbone. We conduct an analysis of the trade-off between reward density and token diversity collapse, demonstrating how continuous feedback structures can greatly affect multi-sample scaling behaviors.

1 Introduction

Reinforcement learning paradigms have become known as successful methods for enhancing the reasoning capabilities of language models Ouyang et al. (2022). By casting token generation as a sequential decision-making process, models can be optimized to maximize an objective instead of only copying token distributions from human demonstrations. Despite these successes, optimizing policies over complex mathematical reasoning problems remains difficult due to the sparse nature of the problem spaces Cobbe et al. (2021); Lewkowycz et al. (2022).

In this work, we analyze the Countdown reasoning task. Given a target number and a set of allowed integers, the model must output a valid algebraic equation (enclosed in structural formatting tags) to reach the target value. The standard baseline evaluation employs a strict scoring system where a token sequence receives a score of 0.0 if structural tags are missing, a structural format score of 0.1 if the syntax is correct but the calculation fails, and a full success score of 1.0 only if the equation uses the exact numbers allowed to accurately match the target value.

2 Related Work

Supervised Fine-Tuning and Preference Optimization Post-training language model alignment has traditionally relied on Supervised Fine-Tuning (SFT) over curated chain-of-thought datasets Wei et al. (2022); Zhang et al. (2024). While SFT establishes formatting compliance and basic capabilities, it can cause models yo be very vulnerable to compounding errors during generation. To alleviate this issue, preference optimization frameworks such as Direct Preference Optimization (DPO) Rafailov et al. (2023) and Identity Preference Optimization (IPO) optimize models directly on chosen and rejected text pairs without requiring an active online reward model.

Online Reinforcement Learning for LLMs To enable continuous self-improvement and exploration beyond fixed datasets, online reinforcement learning algorithms like Proximal Policy Optimization (PPO) Schulman et al. (2017) and Reinforce Leave-One-Out (RLOO) Ahmadian et al. (2024) are often utilized. RLOO optimizes policies directly from online rewards by utilizing a rolling multi-sample baseline. By generating a group of responses for a single prompt and subtracting the average reward of the remaining samples from each individual sequence's reward, RLOO isolates an advantage signal for policy gradient updates.

Reward Shaping and Potential-Based Methods However, online RL methods can be very sensitive to reward design. In traditional reinforcement learning literature, reward shaping is widely recognized as a tool that can be difficult to use. Ng et al. (1999). While dense rewards smooth out complex optimization surfaces, they can cause unintended behaviors if the agent finds optimization shortcuts that maximize reward numbers without fulfilling the main task. Our bounded proximity reward design draws inspiration from potential-based reward shaping theory Ng et al. (1999), with the goal that the shaped rewards preserve the optimal policy of the original sparse reward function while providing denser learning signals.

Mathematical Reasoning with LLMs Recent work on mathematical reasoning has demonstrated the effectiveness of verifier-based approaches Cobbe et al. (2021) and chain-of-thought prompting Wei et al. (2022) for improving arithmetic problem-solving. However, these methods typically rely on static datasets and do not explore online policy optimization. Our work complements this line of research by investigating how reward design affects the dynamics of online reinforcement learning for arithmetic tasks.

Mode Collapse in RLHF The phenomenon of mode collapse during RL training has been documented in recent studies Zhao et al. (2023). Low entropy policies can achieve high single-sample performance but may struggle with multi-sample scaling due to reduced exploration capacity. Our analysis contributes evidence for this trade-off in the specific context of arithmetic reasoning tasks.

3 Method

We formalize our post-training sequence optimization over the Countdown task utilizing an initial policy π_{SFT} and a trainable policy π_θ. Given a prompt x containing a target integer T and an allowed list of numbers N, the model generates a text response y consisting of an equation enclosed within a structural tag block: `<answer>`y_{eq}`</answer>`.

The baseline sparse reward function $R_{sparse}(x, y)$ evaluates generations according to three distinct categorical tiers:

$$R_{sparse}(x,y) = \begin{cases} 0.0 & \text{if structural tags are missing} \\ 0.1 & \text{if syntax is valid but calculation Eval}(y_{eq}) \neq T \\ 1.0 & \text{if syntax is valid and Eval}(y_{eq}) = T \end{cases} \tag{1}$$

To provide an active learning signal for near-miss configurations, we introduce our extension, the Bounded Proximity Reward function $R_{dense}(x, y)$. When an equation is syntactically valid but fails to hit the exact target integer, we compute the absolute numerical distance between the equation's evaluation and the goal target: $\Delta = |\text{Eval}(y_{eq}) - T|$. The shaped reward function is defined as follows:

$$R_{dense}(x,y) = \begin{cases} 0.0 & \text{if structural tags are missing} \\ 0.1 + \frac{\alpha}{1+|\text{Eval}(y_{eq})-T|} & \text{if syntax is valid but Eval}(y_{eq}) \neq T \\ 1.0 & \text{if syntax is valid and Eval}(y_{eq}) = T \end{cases} \tag{2}$$

where α represents a scaling coefficient for the strength of the proximity bonus. Note that the function is bounded so that $0.1 \leq R_{dense}(x, y) < 1.0$ for all incorrect variations. This strict bounding prevents subgoal farming by ensuring that the model cannot accumulate higher expected rewards by repeatedly generating high-proximity errors rather than executing a single flawless calculation path.

Policy updates are calculated within the RLOO framework Ahmadian et al. (2024). For each prompt, a group of k independent completions are sampled from the current policy π_θ. The reinforcement

advantage A_i for the i-th completion is calculated by executing a leave-one-out baseline computation against its peer samples:

$$A_i = R(x, y_i) - \frac{1}{k-1} \sum_{j \neq i} R(x, y_j) \tag{3}$$

This advantage directly scales the policy gradient update, combined with an active token-level KL-divergence penalty against the reference model to prevent catastrophic policy disruption Stiennon et al. (2020).

4 Experimental Setup

Our experimental environment is initialized using a pre-trained $Qwen2.5 - 0.5B$ model backbone Team (2024), namely the provided SFT default checkpoint.

For the reinforcement learning phase, we use the bounded proximity reward function R_{dense} with the proximity hyperparameter set to $\alpha = 0.08$, a value found using a standard grid search.

Both configurations share the same training hyperparameter constants to maintain an accurate baseline control. These include a constant learning rate of 1×10^{-5}, an online KL-divergence penalty weight of $\beta = 0.05$ Stiennon et al. (2020), an effective batch size of 128 prompts, and a generation group size of $k = 4$ per RLOO prompt.

5 Results

5.1 Quantitative Evaluation

Following completion of training, both model policies are evaluated on an independent test dataset of unseen Countdown configurations, namely the provided default evaluation pipeline. From this, we gather the pass@k metric Chen et al. (2021), representing the probability that at least one of k generated samples solves the given input. The results are summarized here in Table 1.

Table 1: Multi-Sample Pass@k Performance Comparison

Method	$pass@1$	$pass@2$	$pass@4$	$pass@8$	$pass@16$
Base RLOO (Sparse)	41.22%	48.00%	53.20%	59.26%	**63.60%**
Dense RLOO (Ours)	**50.38%**	**54.10%**	**57.27%**	**59.71%**	62.00%

Our dense reward model exhibits superior single-turn capability, matching the target on the first attempt in **50.38%** of cases, outperforming the base sparse model's score of **41.22%**. However, as the inference sample count increases, the base RLOO model ultimately climbs to **63.60%** at $pass@16$. In contrast, the Dense RLOO model's curve flattens more quickly, possessing a smaller increase from 50.38% to 62.00%.

5.2 Qualitative Analysis

To understand the dense policy's scaling behavior, we analyze the internal loss and sampling metrics recorded during training.

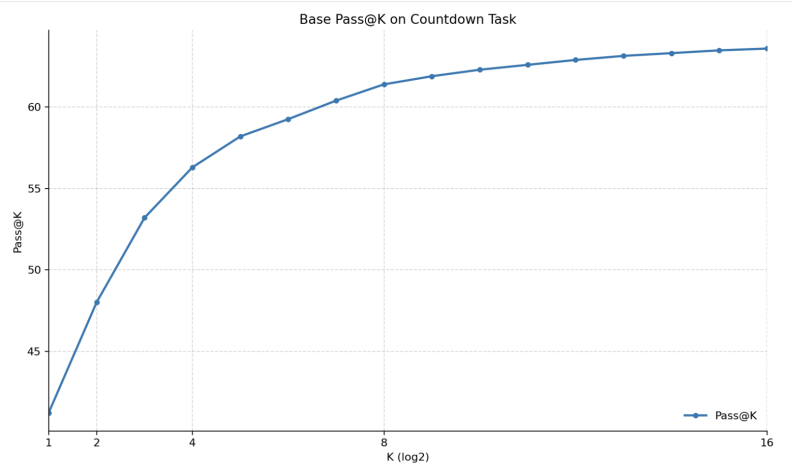


Figure 1: Base RLOO Pass@K scaling on Countdown task. The model scales from 41.22% at $k = 1$ to 63.60% at $k = 16$, indicating good token diversity across multiple samples.

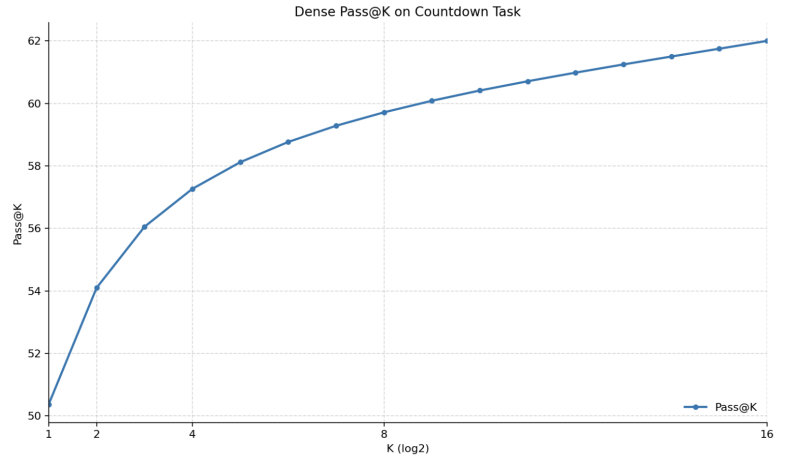
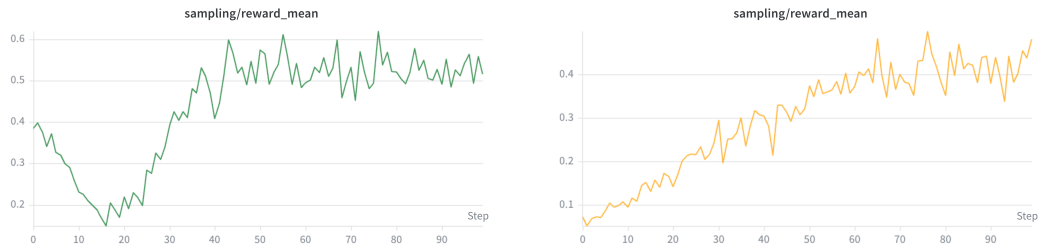


Figure 2: Dense RLOO Pass@K scaling on Countdown task. The model achives 50.38% at $k = 1$ and 62.00% at $k = 16$, indicating reduced token diversity.



(a) Base RLOO Reward

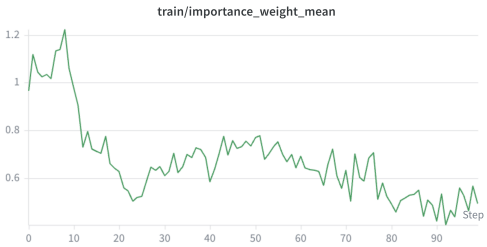
(b) Dense RLOO Reward

Figure 5: Mean reward progression during training. Dense RLOO achieves more consistent reward convergence due to continuous feedback signal.

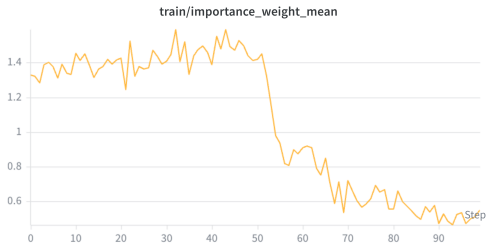
(a) Base RLOO KL Loss

(b) Dense RLOO KL Loss

Figure 3: KL-Divergence comparison between Base and Dense RLOO training.



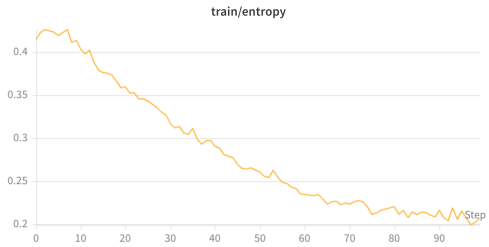
(a) Base RLOO Importance Weight



(b) Dense RLOO Importance Weight

Figure 4: Importance weight dynamics. Dense RLOO shows a sharp drop at step 50.

(a) Base RLOO Entropy



(b) Dense RLOO Entropy

Figure 6: Policy entropy comparison

(a) Base RLOO Train Loss

(b) Dense RLOO Train Loss

Figure 7: Training loss trajectories.

- **KL Divergence Growth Rate:** the policy divergence metrics show that while the Base RLOO run experiences erratic, non-linear fluctuations (peaking near 0.48 as shown in Figure 3a), the Dense RLOO run exhibits a very stable linear KL growth curve that hits a maximum of 0.18 at step 100 (Figure 3b). This confirms that the proximity reward function

presents a clean, low-variance gradient that allows the policy to alter its weights smoothly and consistently.

- **Importance Weight Collapse:** Analysis of the mean importance weight ($\frac{\pi_\theta(a|s)}{\pi_{ref}(a|s)}$) during the training run shows that for the first 50 training steps, the dense importance weight stays around a high baseline of 1.3. However, at step 50, the metric drops sharply to 0.5 (Figure 4b). This indicates that the policy has concentrated its entire probability mass onto a narrow set of learned tokens.
- **Entropy Dynamics:** The entropy plots (Figure 6) reveal hihgly different optimization trajectories. Base RLOO exhibits an initial exploration phase where entropy spikes to ~ 0.9 around steps 15–20, largely caused by high-variance gradients from sparse rewards. This is followed by a sharp collapse near step 40 and decline, eventually reaching a lower final entropy (~ 0.15) than the dense variant. Importantly, this early exploration enables the policy to discover multiple diverse solution pathways before converging, which explains its higher $pass@16$ value. In contrast, Dense RLOO has a more stable entropy decline from ~ 0.42 to ~ 0.2 throughout training. This allows the policy to rapidly converge on a single high-performing generation template, yielding superior $pass@1$ accuracy, but at the cost of reduced exploratory diversity during training.

  Through a qualitative look at some sample trajectories, we noticed that Base RLOO tends to stick to a selection of multiple valid solution templates it discovered during its early exploration but is able to analyze the size of the given numbers. For example, Base RLOO has multiple different ways to start a trajectory, such as:

  - "Let me try to find a way to get to 12 using these numbers. First, let me try to work with the larger numbers to get a smaller number, since 12 is smaller than many of the original numbers."
  - "Let me try to find a way to get to 48 using these numbers. First, let me try to work with the larger numbers 71 and 6:"

  On the other hand, Dense RLOO always begins with "Let me analyze this step by step: 1. First, let's try to get close to target with basic operations:".

Qualitative inspection of the actual evaluation samples confirms the presence of token diversity collapse. When sampled at a standard evaluation temperature, the base RLOO model is able to generate different kinds of trajectories and tokens to reach its goal.

On the other hand, our Dense RLOO model outputs nearly identical token tracks across all 16 generations. For example, the first step across all sampled trajectories always begins with: " 1. First, let's try to get close to 33 with basic operations:". If its primary template contains an error that fails to reach the target, all subsequent samples repeat that identical failure.

6 Discussion

The trade-off between first-turn precision ($pass@1$) and multi-sample exploration ($pass@k$) shows an important design choice for real-world production deployments. In most software, executing multi-sample generations ($k \geq 16$) results in latency and high token costs, so a model that delivers a 50.38% success rate on the first attempt is usually superior and more cost-effective than a model requiring more parallel attempts to find a valid solution. However, for complex reasoning tasks where exploratory diversity is important, token diversity collapse should be avoided.

7 Future Work

One line of future work is to incorporate a better heuristic for measuring the reward a trajectory should receive. One option that I explored but was unsuccessful in was creating a symbolic solver to provide ground truth solutions to evaluate intermediate states. This could help the agent identify which sub-trajectories contain useful examples to learn from to stitch together.

8 Conclusion

In this paper, we investigated the impacts of reward shaping on post-training language model optimization using the Countdown arithmetic task. By combining standard sparse validation rewards with a continuous proximity bonus, we smoothed out the optimization landscape, generating a significant improvement in single-turn capability ($pass@1$ increasing from 41.22% to 50.38%). However, our investigation showed that dense reward tracking creates low-entropy, peaky token distributions that cause mode collapse at higher sampling rates. Future research will focus on providing more accurate reward signals to promote diversity.

9 Team Contributions

- **Ian Chen:** Performed all tasks.

Changes from Proposal: The initial project proposal outlined exploring iterative preference tuning using consecutive stages of IPO fine-tuning. However, initial evaluations revealed dataset variance bottlenecks. Consequently, we updated our project to target a custom dense reward within RLOO training. Later, I pivoted to trying to create a symbolic solver to estimate rewards for intermediate states, but had trouble with formatting intermediate states in LLM responses.

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 preprint arXiv:2402.14740* (2024).

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Subhash Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichi Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention. *arXiv preprint arXiv:2309.06180* (2023).

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving Quantitative Reasoning Problems with Language Models. *arXiv preprint arXiv:2206.14858* (2022).

Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*. 278–287.

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. *arXiv preprint arXiv:2305.18290* (2023).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to Summarize with Human Feedback. *arXiv preprint arXiv:2009.01325* (2020).

Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903* (2022).

Peiyuan Zhang, Ke Sun, Zhiyuan Xu, Sheng Gao, Biqi Pan, Xiaotian Yue, and Bin Li. 2024. The Capacity Emergence of Large Language Models: A Survey. *arXiv preprint arXiv:2402.08706* (2024).

Yiran Zhao, Rishabh Joshi, Tianqi Liu, Madian Khabsa, Fei Fang, and Andrew Howard. 2023. Is Reinforcement Learning (Not) for Natural Language Generation: Benchmarks, Metrics, and Training Objectives for RLHF. *arXiv preprint arXiv:2309.08567* (2023).