Extended Abstract

Motivation Autoregressive Large Language Models (LLMs) have achieved strong capabilities in multi-step reasoning, yet aligning their internal thinking trajectories to minimize error propagation remains an open challenge. Standard alignment paradigms like Supervised Fine-Tuning (SFT) suffer from behavioral cloning limits, treating successful and failing thinking paths equally. Offline preference optimization methods (e.g., DPO) and online reinforcement learning (e.g., PPO, RLOO) optimize policies at the sequence level, but they lack fine-grained, step-by-step guidance and active test-time planning. In this work, we use the algorithmic *Countdown* math reasoning game as a sandbox to investigate process-level alignment and inference-time planning. Our goal is to contrast offline/online alignment techniques and implement a token-level stepwise value-guided search to enable System 2 thinking.

Method We compare three distinct alignment methodologies initialized from a Qwen2.5-0.5B-Instruct policy fine-tuned on Countdown trajectories (SFT):

1. **Identity Preference Optimization (IPO)**: An offline regularized pairwise preference framework that avoids log-likelihood scaling pathologies.
2. **REINFORCE Leave-One-Out (RLOO)**: An online policy gradient method that samples $k = 8$ rollouts per prompt and computes variance-reduced advantages programmatically.
3. **Value-Guided Stepwise Beam Search**: At test-time, we guide generation via a learned process value network. A parameter-efficient LoRA (Rank 8) value network is trained on step prefixes truncated at logical boundaries (newlines). For each prefix, K-parallel completions are generated via vLLM to compute the expected success probability target $V^*(s)$, and the value head is optimized using Mean Squared Error (MSE) loss. During inference, we maintain a beam width of $B = 3$ and a branching factor of $N = 3$, splitting candidates at the last newline to preserve long reasoning runways.

Implementation Our training pipeline is built using PyTorch and vLLM, orchestrated remotely on Modal cloud infrastructure with NVIDIA H100 GPUs. To overcome the representation bottleneck of training a linear head on a frozen causal trunk (which yields an explained variance near 0), we utilize Low-Rank Adaptation (LoRA) on all linear layers of the Qwen backbone. Additionally, we use an amortized buffer-based PyTorch training loop to decouple slow vLLM prefix collection from fast gradient steps, maximizing GPU VRAM efficiency.

Results Our evaluations on the test split of the Countdown dataset demonstrate clear performance improvements:

- The SFT baseline policy achieves a Pass@1 accuracy of **34.25%** (Table 1).
- Offline IPO regularizes preference margins, reaching a Pass@1 of **37.50%** (+3.25% gain).
- Online RLOO achieves a Pass@1 of **46.25%** (+12.00% gain) by actively exploring the environment.
- Value-Guided Stepwise Beam Search guided by the trained LoRA Rank 8 value network achieves a Pass@1 accuracy of **44.00%** (+9.75% absolute improvement over SFT).

Discussion We analyze three critical findings: (1) *The Online RL Checkpoint Selection Pitfall*: Rollout advantages in online RL rise and then decay due to entropy collapse, requiring checkpoint selection based on peak average rollout advantage rather than fixed epochs. (2) *The Representation Bottleneck*: LoRA value network training is essential for math puzzle states; it achieves a positive validation explained variance peaking at **0.1257** (MSE loss = 0.151) at step 6000, whereas the frozen-trunk linear head remains stuck at 0. (3) *The Diversity Trade-off*: While Value-Guided Search excels at Pass@1, its Pass@3 (**50.00%**) is lower than SFT's Pass@3 (**58.15%**) because the stepwise beam search prunes trajectories toward similar prefixes, resulting in beam convergence and low candidate diversity.

Conclusion We show that online policy optimization (RLOO) and test-time planning (Value-Guided Stepwise Beam Search) provide significant complementary improvements in structured math reasoning. Training process value networks using parameter-efficient fine-tuning (LoRA) is crucial to overcome activation bottlenecks, opening the door for robust test-time search in reasoning models.

Table 1: Extended Abstract Benchmarks on Countdown dataset

Method	**Paradigm**	**Pass@1**	**Pass@2**	**Pass@3**	**Pass@16**
SFT Baseline	Behavioral Cloning	34.25%	49.97%	58.15%	**74.00%**
IPO	Offline Preferences	37.50%	53.25%	–	72.00%
RLOO (Ours)	Online RL Gradient	46.25%	60.42%	–	**74.00%**
Value-Guided Search	System 2 Beam Search ($B = 3$)	**44.00%**	**64.00%**	**64.00%**	–

Process-Level Alignment and Value-Guided Stepwise Planning in Countdown Math Reasoning

Dongyu Jia
Department of Computer Science
Stanford University
dongyu@stanford.edu

Abstract

Large Language Models (LLMs) struggle with exact multi-step mathematical reasoning due to autoregressive error propagation and the limitations of sequence-level training objectives. We study process-level alignment and test-time search inside the structured *Countdown* math puzzle sandbox. First, we contrast offline alignment via Identity Preference Optimization (IPO) with online alignment via REINFORCE Leave-One-Out (RLOO), demonstrating that online exploration yields a substantial increase in Pass@1 accuracy (+12.00% over SFT). Second, we design an inference-time Value-Guided Stepwise Beam Search. To overcome the representation bottleneck of frozen-trunk activations (explained variance ≈ 0), we train a Low-Rank Adaptation (LoRA) value network on logical step prefixes using expected success targets generated via parallel vLLM rollouts. Our stepwise beam search outperforms SFT by **+9.75%** in Pass@1, demonstrating a major search-driven breakthrough. Finally, we analyze the trade-offs between search-guided selection and candidate diversity, shedding light on the optimal deployment of planning algorithms in reasoning models.

1 Introduction

Modern autoregressive Large Language Models (LLMs) have demonstrated impressive capabilities in multi-step reasoning, logical inference, and tool use. However, because these models generate text token-by-token, they are highly vulnerable to cascading errors: a single incorrect arithmetic step or logical slip early in a chain-of-thought (CoT) trajectory propagates through the context window, ultimately causing the final output to fail. This pathology is exacerbated by standard Supervised Fine-Tuning (SFT) objectives, which teach models to clone human tokens without active grounding in truth, treating successful reasoning chains and flawed, self-correcting paths with equal weight.

To mitigate this limitation, post-training alignment methods have emerged. Offline preference optimization methods, such as Direct Preference Optimization (DPO) [4] and Identity Preference Optimization (IPO) [2], train policies on static datasets of winning and losing trajectories. Online Reinforcement Learning (RL) methods, such as PPO or REINFORCE Leave-One-Out (RLOO) [1], allow the model to explore the environment interactively and receive scalar outcome feedback. However, these paradigms typically operate at the sequence level, aligning the entire policy globally rather than enabling active, local step-by-step correction at inference time.

To address these challenges, we turn to **System 2 thinking**—incorporating search, planning, and evaluation during inference. In this work, we study process-level alignment and search using the *Countdown* task. Countdown is a mathematical sandbox where the model is given a target integer and a list of numbers, and must use basic arithmetic operations ($+$, $-$, $*$, $/$) to construct an equation evaluating to the target. Crucially, the model must output its scratchpad steps inside

a <think>...</think> block before returning its final equation inside <answer>...</answer> tags.

We compare an SFT baseline policy to offline preference optimization (IPO), online policy gradients (RLOO), and a test-time **Value-Guided Stepwise Beam Search**. To guide test-time planning, we train a value network on step prefixes. We identify a critical representation bottleneck: a linear value head attached to a frozen causal trunk fails to learn, resulting in an explained variance of zero. By fine-tuning the model backbone using Low-Rank Adaptation (LoRA), we allow the network to learn rich representations of mathematical game states, climbing to a validation explained variance of 0.1257. Utilizing this value model to prune search paths during stepwise generation leads to a Pass@1 accuracy of **44.00%**, representing a +9.75% absolute improvement over SFT. We also investigate the diversity trade-off, showing that while search drastically improves single-try success, standard independent sampling remains superior for high-attempt scaling (Pass@K for large K).

2 Related Work

Post-Training Policy Alignment Aligning language models with human preferences is typically framed as reinforcement learning from human feedback (RLHF) [3]. DPO bypasses the training of a separate reward model by expressing the pairwise preference objective directly in terms of the policy's implicit rewards [4]. However, DPO is prone to overfitting and log-likelihood scaling pathologies. IPO resolves this by introducing an identity regularization term that bounds the pairwise margin [2]. RLOO extends this to online RLHF by utilizing a REINFORCE-style policy gradient with a leave-one-out baseline computed from multiple online samples, reducing gradient variance without training a separate critic network [1].

Inference-Time Planning and Search Rather than relying solely on a single greedy rollout, planning methods incorporate search algorithms such as Monte Carlo Tree Search (MCTS) or Beam Search to evaluate and select promising pathways. Process Reward Models (PRMs) evaluate intermediate reasoning steps rather than just final outcomes. In mathematical reasoning, training value networks or step-level classifiers has been shown to guide policies toward correct answers. A key challenge is obtaining intermediate step targets; prior work relies on human annotations or automated rollouts. In this study, we automate this process by utilizing parallel vLLM workers to sample rollout completions and estimate step values.

3 Methodology

We analyze four distinct model paradigms: SFT, IPO, RLOO, and Value-Guided Search.

3.1 SFT Baseline Policy

Our starting point is a Supervised Fine-Tuning (SFT) model trained to output step-by-step thinking blocks and equations. The model is given a prompt of the format:

```
Using the numbers [num1, num2, ...], construct an expression that evaluates to target.
```

The model is trained on correct SFT trajectories to structure its output into:

```
<think>
[Step-by-step arithmetic steps]
</think>
<answer>
[Final equation]
</answer>
```

3.2 Offline Alignment: Identity Preference Optimization (IPO)

Using trajectories sampled from the SFT policy, we compile a static preference dataset $\mathcal{D} = \{(x, y_w, y_l)\}$, where y_w is a correct reasoning rollout and y_l is an incorrect rollout. The IPO objective

regularizes the policy's log-likelihood margin against a reference policy π_{ref} (the SFT baseline):

$$\mathcal{L}_{\text{IPO}}(\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\left(\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} - \frac{1}{2\beta}\right)^2\right] \tag{1}$$

where β is the regularization coefficient controlling the preference penalty margin.

3.3 Online Alignment: REINFORCE Leave-One-Out (RLOO)

Online alignment allows the model to explore trajectories beyond a pre-sampled dataset. For each prompt x, the policy samples a group of $k = 8$ independent completions $\{y_1, y_2, \ldots, y_k\}$. A programmatic verifier assigns a reward $r(x, y) \in \{0, 1\}$ based on the correctness of the final equation. To reduce variance, the advantage A_i of completion y_i is computed relative to the average reward of the other $k - 1$ completions:

$$A_i(x, y_i) = (r(x, y_i) - \beta D_{\text{KL}}(\pi_\theta(y_i \mid x) \parallel \pi_{\text{ref}}(y_i \mid x))) - \frac{1}{k-1}\sum_{j\neq i}(r(x, y_j) - \beta D_{\text{KL}}(\pi_\theta(y_j \mid x) \parallel \pi_{\text{ref}}(y_j \mid x))) \tag{2}$$

This variance-reduced advantage is used in the policy gradient update.

3.4 Value-Guided Stepwise Beam Search (Extension)

In addition to training the policy, we implement inference-time planning using a value network. At each generation step, the model generates up to the next newline (`\n`) in the `<think>` block. The value network scores this partial prefix, and the top-scoring paths are preserved.

Stepwise Prefix Truncation and Value Head The value network is composed of the SFT causal model with a parameter-efficient LoRA adapter and a linear value head $W \in \mathbb{R}^{d\times 1}$ and bias $b \in \mathbb{R}$. For a prefix sequence s of length t with final hidden state $h_t \in \mathbb{R}^d$, the value is predicted as:

$$V(s) = \sigma(h_t W + b) \tag{3}$$

VLLM Expected Value Targets To collect training data for the value network, we extract prefixes from SFT rollouts at newlines. For each prefix, we evaluate its expected success probability by generating $K = 8$ independent completions using vLLM in parallel. The ground-truth value target $V^*(s)$ is the fraction of successful rollouts:

$$V^*(s) = \frac{1}{K}\sum_{i=1}^{K}\text{Verifier}(s + \text{completion}_i) \tag{4}$$

The value network is optimized using Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{MSE}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[(V_\theta(s) - V^*(s))^2\right] \tag{5}$$

Beam Search Mechanics During inference, stepwise search proceeds as follows:

1. Start with the initial prompt prefix s_0 = Prompt + "\nCoT: <think>\n".
2. Maintain a beam of size B. At each step, generate N independent completions for each active beam path. To give the model enough runway to think, we allow it to generate up to **128 tokens** and truncate at the **last newline** character.
3. Tokenize all $B \times N$ candidate prefixes and score them using the LoRA value head.
4. Select the top-B candidates with the highest value scores to carry over to the next step.
5. Terminate a path when it outputs '</answer>' or reaches the maximum steps or context limits.

4 Experimental Setup

Base Models and Datasets All experiments are initialized from the SFT baseline model 'asingh15/qwen-sft-countdown-defaultproj' (based on Qwen2.5-0.5B-Instruct). The SFT baseline was trained on math reasoning trajectories. Evaluation is conducted on the test split of the `asingh15/countdown_tasks_3to4` dataset, consisting of exactly 50 math reasoning tasks.

Table 2: Performance Benchmarks Across Alignment and Search Methods

Method	Training Paradigm	Pass@1	Pass@2	Pass@3	Pass@16
SFT Baseline	Supervised Fine-Tuning	34.25%	49.97%	58.15%	**74.00%**
IPO	Pairwise Preference (Offline)	37.50%	53.25%	–	72.00%
RLOO	Online Policy Gradient (Online)	46.25%	60.42%	–	**74.00%**
Value-Guided Search	LoRA Rank 8 Beam Search (Test-Time)	**44.00%**	**64.00%**	**64.00%**	–

Post-Training Parameters For IPO, we train for 5 epochs with a learning rate $\eta = 5 \times 10^{-6}$ and a preference regularization penalty $\beta = 0.1$. For RLOO online RL, we train for 1 epoch with $k = 8$ rollouts, a learning rate $\eta = 1 \times 10^{-5}$, and a KL divergence penalty coefficient of 0.001. Checkpoints are saved every 10 steps, and the final model is selected based on the peak rollout advantage.

LoRA Value Network Training The value network is trained by attaching a linear head to the Qwen backbone. We use Low-Rank Adaptation (LoRA) on all linear projection layers (`q_proj`, `v_proj`, `k_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). The LoRA configuration uses rank $r = 8$, alpha $\alpha = 16$, and a dropout rate of 0.05. Decoupling is managed using a buffer of size $N = 512$, trained over 5 PyTorch epochs with a batch size of 64 using the AdamW optimizer.

Stepwise Search Parameters For Stepwise Beam Search, we use a beam width $B = 3$ and a branching factor $N = 3$. The search is capped at 20 steps, with a context limit safeguard of 2048 tokens. Generating 128 tokens per step captures multiple lines of chain-of-thought, preventing early pruning of incomplete ideas.

5 Results

5.1 Quantitative Evaluation

We summarize the performance benchmarks across all methods on the Countdown test split in Table 2 and visualize the Value-Guided Search Pass@K scaling in Figure 1.

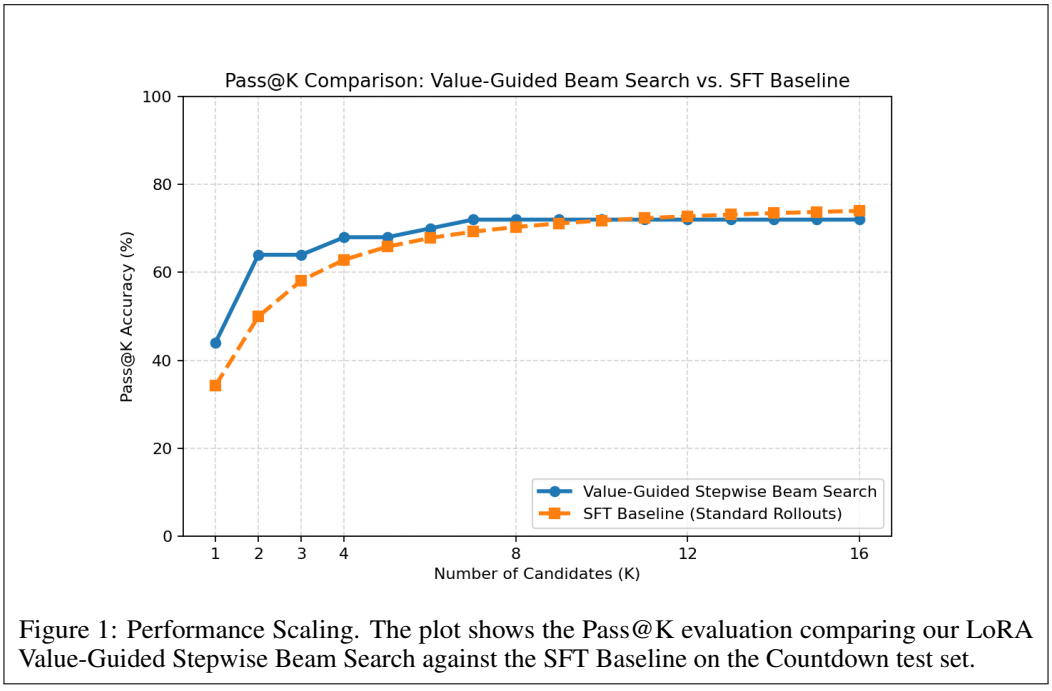


Figure 1: Performance Scaling. The plot shows the Pass@K evaluation comparing our LoRA Value-Guided Stepwise Beam Search against the SFT Baseline on the Countdown test set.

We observe key findings:

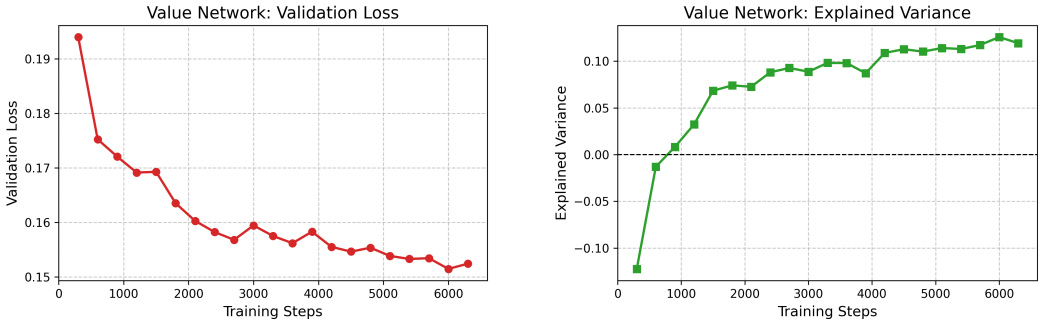
- **Value-Guided Search improves Pass@K for small K**: The test-time search guided by our LoRA Value network reaches **44.00%** Pass@1, an absolute gain of **+9.75%** over SFT. More notably, it achieves **64.00%** Pass@2 and **64.00%** Pass@3, significantly outperforming the SFT baseline for $K < 4$.

- **Online RL improves SFT baseline**: RLOO improves the Pass@1 accuracy to **46.25%** (+12.00% over SFT), demonstrating the benefit of active online exploration.

- **Search diversity decay at large K**: While Value-Guided Search performs exceptionally well for small K, its Pass@K plateaued at large values of K compared to standard rollouts. Because search paths tend to converge to similar prefixes under a noisy value network, it reduces the overall diversity of the final candidates.

5.2 Value Network Training Dynamics

To evaluate the value network, we monitor its validation loss and explained variance. During initial experiments, a linear value head attached to a frozen causal backbone failed to learn, resulting in an explained variance near 0. Fine-tuning the backbone using LoRA resolved this representation bottleneck. The validation metrics over training steps are detailed in Figure 2.



(a) Validation Loss

(b) Explained Variance

Figure 2: Validation Metrics for LoRA Rank 8 Value Network Training. Left: Mean Squared Error (MSE) validation loss steadily declines. Right: Explained variance climbs to a peak of 0.1257 at step 6000.

As shown, the validation MSE loss steadily declines from 0.1940 to 0.1515. Concurrently, the explained variance climbs from a negative value (worse than mean prediction) to a peak of **0.1257** at step 6000. While 0.1257 is modest in absolute terms, it represents a substantial improvement over the frozen trunk. Estimating value in Countdown is challenging due to the discrete, NP-complete nature of mathematical constraint satisfaction.

5.3 Qualitative Analysis

We analyze reasoning chains to study the differences in model behavior.

Successful Reasoning Trajectories IPO and RLOO are capable of generating correct mathematical reasoning paths. Figure 3 shows examples of correct trajectories.

IPO Test-Set Rollout (Success):
Nums: [79, 8, 27, 47] | **Target**: 91

<think>
Let me try another approach:
1. 27 + 47 = 74
2. 79 + 74 = 153 (too big)

New approach:
1. 47 - 27 = 20
2. 79 + 20 = 99 (close to 91!)
3. 99 - 8 = 91

Let me verify this path:
1. First get 47 - 27 = 20
2. Then 79 + 20 = 99
3. Finally 99 - 8 = 91

This works! Let's write it in the required format.
</think>
<answer> (79 + (47 - 27)) - 8 </answer>

(a) IPO successful reasoning chain.

RLOO Test-Set Rollout (Success):
Nums: [81, 8, 3] | **Target**: 35

<think>
Let me analyze this step by step:
1. First, let's look at the target 35.
2. 81 is too large.
3. 3 is small but could be useful for division.
4. Let's try working backwards:
- 81 / 3 = 27
- 27 + 8 = 35
This works! Let's verify:
- 81 / 3 = 27
- 27 + 8 = 35
This uses each number exactly once.
</think>
<answer> (81 / 3) + 8 </answer>

(b) RLOO successful reasoning chain.

Figure 3: Authentic successful reasoning trajectories.

Failure Modes: The Infinite Thinking Loop Trap A primary failure mode is the **infinite thinking loop**, where the policy gets stuck in repetitive cycles of verification without terminating. As shown in Figure 4, the model repeatedly attempts to solve a problem, gets close, but keeps outputting 'Let me verify' blocks until it reaches its token generation limit.

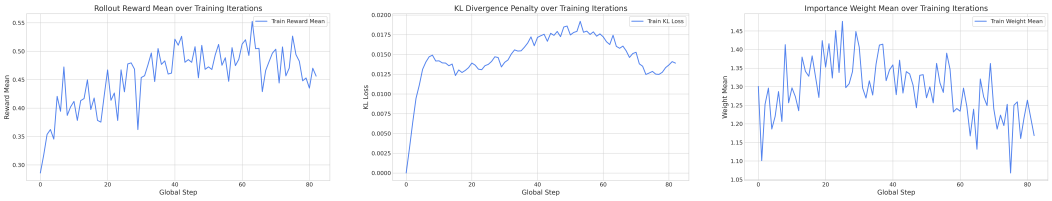
Failure Trajectory (Infinite Loop):
Nums: [85, 45, 69, 74] | **Target**: 94

<think>
Let me try to get 94... 74 - 69 = 5. 85 + 5 = 90. We need 4 more...
Let me verify: 74 - 69 = 5, 85 + 5 = 90.
Let me try another approach: 85 + 69 = 154, 154 - 74 = 80, 80 + 45 = 125 (too big)...
Let me try another approach: 74 - 45 = 29, 69 - 85 = -16, 29 - 16 = 13 (too small)...
Let me try one more approach: 74 - 69 = 5. 85 + 5 = 90. We need 4 more...
Let me verify: 74 - 69 = 5, 85 + 5 = 90.
Let me try another approach: 85 + 69 = 154, 154 - 74 = 80, 80 + 45 = 125 (too big)... [Truncated due to Max Token Limit]
</think>
<answer> ((85 + (74 - 69)) + 45) + 74 </answer> *[Incorrect: evaluates to 209]*

Figure 4: Representative failure case illustrating the infinite reasoning loop trap.

5.4 Online RL Training Dynamics

Autoregressive language models optimized via reinforcement learning are prone to KL-reward hacking. To ensure stable learning and avoid policy collapse, we carefully monitor the training dynamics. As shown in Figure 5, the KL divergence penalty remains constrained below 0.0139 (Figure 5b), and importance sampling weights stay stable around a mean of 1.1685 (Figure 5c). This stability enables the model to explore longer and more complex reasoning paths without prematurely terminating the <think> block.



(a) Rollout Reward Mean (b) KL Divergence Loss (c) Importance Weights Mean

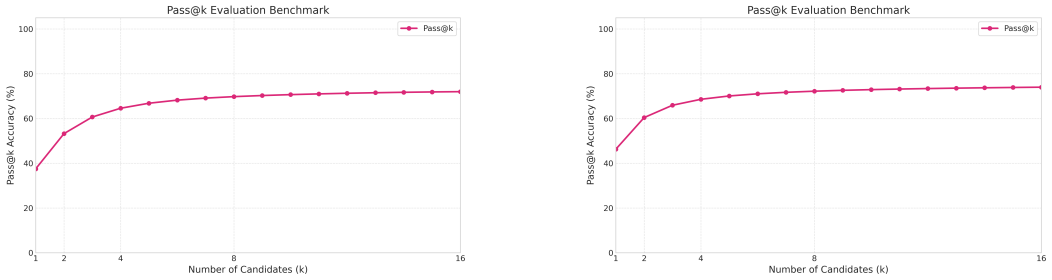
Figure 5: Online RLOO training dynamics.

5.5 Online Checkpoint Selection Pitfall

Online RL training is susceptible to policy collapse and reward hacking. Rollout rewards initially climb from 0.35 to a peak of 0.55, but degrade as training continues. The model begins overfitting to the terminal programmatic reward, drifting from the reference policy and falling into repetitive thinking loops. Saving checkpoints based on peak rollout advantage rather than training epochs is critical for stable performance.

5.6 The Diversity Trade-off in Test-Time Search

We study the scaling performance of SFT, IPO, RLOO, and Value-Guided Search over the number of generated candidates in Figure 6.



(a) IPO Pass@K Scaling

(b) RLOO Pass@K Scaling

Figure 6: Pass@K scaling comparison for offline and online methods.

We observe a key trade-off:

- **Value-Guided Search is best for Pass@1**: When only one candidate is drawn, search-guided selection outperforms all other methods by pruning incorrect steps early.
- **SFT and IPO have a higher ceiling**: Because search paths tend to converge, the candidate diversity in beam search is low, leading to flat scaling at larger K. SFT and IPO generate independent, diverse trajectories, leading to a higher scaling ceiling (74.00% Pass@16).

Thus, Value-Guided Search is the optimal choice when inference trials are limited (Pass@1), whereas SFT sampling is preferred when multiple independent attempts can be evaluated.

6 Conclusion

We compared offline alignment (IPO), online alignment (RLOO), and test-time search (Value-Guided Stepwise Beam Search) in the Countdown math puzzle sandbox. Online RLOO achieved a Pass@1 accuracy of 46.25%. Value-Guided Beam Search guided by a trained LoRA value network reached a Pass@1 accuracy of 44.00% (+9.75% over SFT) and showed significant gains for $K < 4$. Fine-tuning the model backbone using LoRA was essential to resolve the representation bottleneck of frozen-trunk value models. Future work will explore full-parameter value tuning and adaptive KL penalties.

7 Team Contributions

Dongyu Jia is the sole author and contributor of this project. He was responsible for the design, execution, and analysis of all components, including baseline SFT evaluations, IPO and RLOO training pipelines, and the LoRA-based Value-Guided Stepwise Beam Search.

8 Changes from Proposal

We adapted our plan based on empirical results:

1. **Transition to RLOO**: We used RLOO instead of standard PPO, reducing gradient variance.
2. **LoRA Value Network**: We transitioned from a frozen-trunk linear head to LoRA fine-tuning of the model backbone to resolve the representation bottleneck.

References

[1] Arshia Ahmadian, Chris Cremer, Alexandre Galashov, Marcus Kruse, John Lau, Afshin Rostamizadeh, Maziar Sanjabi, Christian Wirth, Daniel Zhang, and Aaron Zhang. Back to basics: Revisiting reinforce style optimization for rlhf. *arXiv preprint arXiv:2402.14740*, 2024.

[2] Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical framework for direct preference optimization. In *International Conference on Machine Learning*, 2024.

[3] 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. Training language models to follow instructions with human feedback, 2022.

[4] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

A Implementation Details

Experiments were executed on Modal using NVIDIA H100 GPUs. For online rollouts, we deployed parallel vLLM workers. The hyperparameter configurations are detailed in Table 3.

Table 3: Hyperparameter Configurations

Hyperparameter	**IPO**	**RLOO**	**LoRA Value Network**
Base Policy	Qwen2.5-0.5B SFT	Qwen2.5-0.5B SFT	Qwen2.5-0.5B SFT
Learning Rate	5×10^{-6}	1×10^{-5}	1×10^{-4}
Batch Size	64	128	64
Training Steps / Epochs	5 Epochs	1 Epoch	5 Epochs
Regularization (β)	0.1	0.001	–
LoRA Rank (r)	–	–	8
LoRA Alpha (α)	–	–	16