Extended Abstract

Motivation Large language models (LLMs) are increasingly equipped with tools like calculators to solve complex reasoning tasks. However, LLMs, especially compact models, often struggle with multi-hop tool use due to the long-horizon credit assignment problem—where it's unclear which intermediate steps contribute to success. This project explores whether reward densification techniques can enhance tool use in such settings. Specifically, we fine-tune Qwen2.5-0.5B on the Countdown math reasoning task. While the model demonstrates strong reasoning structure after supervised fine-tuning and Reward Learning from Observations (SFT+RLOO), it still suffers from low accuracy due to arithmetic errors. This motivates the integration of tool use (calculator calling) to improve performance.

Method Our extension modifies SFT training with an augmented dataset that demonstrates tool usage, and we experiment with the novel reward densification framework we designed - implicit and explicit - during RLOO training. Specifically, we explore the following:

- **Baseline (SFT + RLOO):** Standard supervised fine-tuning followed by sparse reward learning using RLOO. We apply cosine learning rate scheduling with warmup, monitor accuracy during training and evaluation, and employ early stopping to prevent hallucination.
- **Tool Calling Enhancements:** Inspired by the Search R1 framework, we incorporate multi-turn tool use through the following mechanisms:
  - **SFT with Tool Tags:** We wrap equations with `<calculator>...</calculator>` tags to teach the model when and how to invoke the calculator.
  - **RLOO with Tool Execution:** During training rollouts and inference generation, whenever a `</calculator>` token is produced, the model pauses. The enclosed expression is evaluated by an external calculator, and the result is injected into the output stream before generation resumes.
- **Reward Densification Strategies:**
  - **Heuristic Reward Shaping:** Assigns intermediate rewards based on interpretable criteria, such as numerical proximity to the target or correct invocation of tools.
  - **Implicit Process Reward Modeling (Implicit PRM):** Implicitly learns dense, token-level rewards from trajectories using log-probability ratios, encouraging step-by-step correctness.
  - **Hindsight Experience Replay:** Implicitly augments training with artificial positive examples by modifying the problem target to match the model's predicted output.

Implementation We perform supervised fine-tuning (SFT) with 1000 samples and reinforcement learning (RL) with 200–400 samples. To reduce overfitting and hallucination, we employ early stopping and learning rate scheduling. We implement a custom calculator tool supporting basic arithmetic with parentheses, and adapt all three reward densification strategies for the Countdown task.

Results Experiments show that heuristic shaping alone is ineffective, likely due to the model's small size. Implicit strategies yield notable gains independently, and combining them with heuristic shaping improves training stability. HER combined with reward phasing works yields an average evaluation reward score of 0.437 and producing 100% well-formed equation during evaluation. We are positive that such results can generalize to other tools and other models of similar size.

Discussion We find that starting RL with a tool-focused heuristic reward, then shifting to answer-focused rewards, improves tool use and final accuracy. This supports reward phasing—beginning with exploration and shifting to exploitation—as a promising paradigm for tool-based alignment. While these results are promising for compact models, generalization to larger models remains uncertain.

Conclusion Implicit reward densification can significantly improve tool use performance in compact LLMs, without requiring handcrafted signals. Combining implicit methods with heuristic shaping further improves training robustness and stability. Future work should formalize reward densification specific for the tool-calling task, and explore reward strategies generalizable across tools.

Reward Densification For RL in Multi-hop Calculators

Coco Xu
Department of Computer Science
Stanford University
cocozxu@stanford.edu

Jenny Chen
Department of Computer Science
Stanford University
jennycjx@stanford.edu

Yolanda Wang
Department of Computer Science
Stanford University
wangyy@stanford.edu

Abstract

Multi-hop tool use remains challenging for compact language models due to long-horizon credit assignment, where it is unclear which intermediate actions contribute to success. We investigate whether reward densification can improve tool use in compact language models, focusing on the Countdown math reasoning task. Using Qwen2.5-0.5B, we integrate a calculator tool and explore three reward strategies: heuristic shaping, implicit process reward modeling (PRM), and hindsight experience replay (HER). Our results show that implicit methods significantly improve performance without handcrafted signals, while combining them with heuristic rewards enhances training stability. We further find that phasing rewards—from tool-focused shaping to accuracy-focused rewards—leads to better tool use and final task performance. These findings give insights to future reward designs for tool use in compact LLMs.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in tasks requiring natural language understanding, generation, and reasoning. However, their performance often worsen when faced with complex multi-step reasoning problems that require precise arithmetic or symbolic manipulation. To address this, recent work has introduced tool-augmented LLMs that can invoke external tools—such as calculators, search engines, or code interpreters—during inference to enhance accuracy and reliability. While promising, these models still struggle with learning when and how to call tools effectively, especially in settings where task supervision is limited or sparse.

One key challenge is the long-horizon credit assignment problem: it is difficult for the model to infer which intermediate reasoning steps are responsible for eventual success or failure. Reinforcement learning from sparse rewards often fails to provide sufficient training signal to refine fine-grained behavior, such as correct tool invocation in intermediate steps. This motivates the use of *reward densification*—techniques that augment sparse reward signals with richer, more informative feedback to accelerate and stabilize learning.

Prior work on reward densification includes potential-based reward shaping Ng et al. (1999), which augments rewards using a potential function without altering the optimal policy; Hindsight Experience Replay (HER) Andrychowicz et al. (2017), which re-labels failed trajectories to simulate successful ones; and Implicit Process Reward Modeling (Implicit PRM) Yuan et al. (2024), which leverages log-likelihood ratios to provide token-level guidance based on outcome alignment. However, existing techniques are largely agnostic to the nuances of tool usage and fail to address the unique challenges of multi-hop reasoning tasks involving external operations like calculator calls.

In this paper, we propose a framework for improving tool-augmented reasoning in LLMs through specialized reward densification techniques. Our contributions are threefold: (1) we fine-tune a compact LLM (Qwen2.5-0.5B) on the Countdown arithmetic reasoning task and show that vanilla supervised fine-tuning (SFT) followed by reward learning from observations (RLOO) yields promising but error-prone results; (2) we introduce tool-aware enhancements, including supervised tool-tagging, inference-time tool execution, and implicit + heuristic reward shaping tailored for tool use; and (3) we conduct extensive experiments demonstrating that our reward-densified framework significantly improves both convergence speed and final performance, even from less satisfactory initial models.

Our findings highlight the importance of task-specific reward shaping in enabling robust multi-step tool use and suggest a broader role for densified feedback in structured language model training.

2 Related Work

Our research extends prior work in three key domains: reinforcement learning for LLMs, tool-calling capabilities, and reward densification techniques.

2.1 Reinforcement Learning for LLM

Reinforcement learning has proven effective for sequential decision making in various domains prior to its application to language models. Foundational approaches like Supervised Fine-Tuning (SFT) Ouyang et al. (2022) and Reinforcement Learning from Human Feedback (RLHF) Ouyang et al. (2022) enabled capabilities seen in models like ChatGPT, with subsequent research improving algorithm efficiency through methods like Direct Preference Optimization (DPO) Rafailov et al. (2023). Despite these advancements, hallucination remains a challenge, motivating research into tool-calling as a mitigation strategy.

2.2 Tool Calling

Tool-calling integration enables LLMs to invoke external resources during reasoning processes to generate better and more accurate results. Early approaches like Toolformer Schick et al. (2023) established the foundation through self-supervised learning, while Search-R1 Jin et al. (2025) refined these capabilities using RL, so that models can learn optimal usage strategies through outcome-based feedback. Current limitations include Toolformer's focus on single API calls when real-world tasks often require multiple tool interactions, and Search-R1's application only to factual reasoning with objective ground truth. Our work extends these approaches to tasks involving multiple reasoning steps (CountDown) and scenarios without single correct answers (UltraFeedback).

2.3 Reward Densification

Reward densification has the benefit of accelerating convergence. Existing reward densification methods include potential-based reward shaping Ng et al. (1999), which adds auxiliary rewards based on state potentials while preserving optimal policies; Implicit Process Reward Modeling (Implicit PRM) Yuan et al. (2024), which derives token-level guidance from outcome labels using log-likelihood ratios between models; and Hindsight Experience Replay (HER) Andrychowicz et al. (2017), which augments learning by reinterpreting failed episodes as successful ones through goal relabeling. Despite extensive research in this area, there remains a gap in reward shaping specifically for effective tool usage—a gap our research addresses through specialized reward mechanisms for tool-calling scenarios in multi-step reasoning tasks without objective ground truth.

Our contribution is a novel reward densification framework that integrates tool calling to improve math reasoning for LLM. No prior work has systematically explored reward shaping for tool-augmented, multi-step mathematical reasoning.

3 Method

We aim to improve tool-augmented reasoning in large language models through a combination of supervised fine-tuning and reinforcement learning with reward densification. Our approach consists of three stages: (1) supervised fine-tuning to establish baseline reasoning structure; (2) tool usage integration at both training and inference stages; and (3) reinforcement learning with reward densification to refine the model's behavior on complex, multi-step reasoning tasks.

3.1 Supervised Fine-Tuning

We begin with supervised fine-tuning (SFT) of the Qwen2.5-0.5B language model on the Countdown dataset—a math reasoning task that requires generating a sequence of arithmetic operations to reach a target number from a given set of operands. Each training example is annotated using `<think>` and `<answer>` tags to teach the model how to structure intermediate reasoning and final answers. The dataset is preprocessed to include both successful and partially correct attempts, allowing the model to learn diverse reasoning patterns.

3.2 Tool Usage Integration

To enable effective calculator tool use, we introduce enhancements to both the training data and the inference pipeline.

3.2.1 SFT with Tool Tags

We augment the SFT data by wrapping expressions that should be evaluated externally in `<calculator>`...`</calculator>` tags. This explicitly teaches the model when to delegate computation to a calculator rather than perform arithmetic internally. For example:

```
<calculator> (75 - 3) * 2 </calculator>
```

3.2.2 Inference-Time Tool Execution

At inference time, when the model generates a `</calculator>` token, generation is paused, the expression inside the most recent `<calculator>` block is extracted and evaluated using a custom calculator module. The result is injected back into the generation stream, and decoding resumes. This mechanism allows the model to defer precise computation while maintaining control flow over the solution process.

3.3 Reinforcement Learning with Reward Densification

After SFT, we further train the model using Reward Learning from Observations (RLOO) with several reward shaping techniques. The objective is to refine tool usage behavior and reduce arithmetic or reasoning errors that persist after supervised training.

3.3.1 Implicit Process Reward Modeling (Implicit PRM)

We implement Implicit PRM Yuan et al. (2024) to provide token-level rewards based on the likelihood ratio between the trained model and a reference model (e.g., an earlier checkpoint). This encourages trajectories that increase alignment with correct outputs without requiring dense supervision.

$$r_t = \beta \log \frac{\pi_\theta(y_{\leq t})}{\pi_{\text{ref}}(y_{\leq t})} \tag{1}$$

3.3.2 Heuristic Reward Shaping

We introduce a multi-component heuristic reward function that provides dense feedback on formatting, tool usage, and mathematical accuracy.

Our reward function combines structural, mathematical, and efficiency components:

Table 1: Heuristic Reward Shaping Components

Reward Type	Component	Purpose	Value
Structural	δ_{ans}	Proper <answer> tags	$+1.0$ / -1.0
	δ_{inv}	Invalid equation penalty	-0.5
Tool Usage	δ_{calc}	Valid calculator usage	$+0.02$ per call, 0.2 max
	δ_{rep}	Redundancy penalty	-0.02 per repeat
	δ_{dc}	Calculator usage bonus	$+0.1$
Mathematical	δ_{fmt}	Correctness bonus	$+2.0$
	δ_{close}	Proximity reward	0 to $+0.4$
	δ_{conc}	Conciseness equations	$+0.2$

Structural Rewards: We reward proper answer formatting and penalize invalid equations that use incorrect numbers or violate problem constraints (use numbers outside of input or use input number multiple times).

Tool Usage Efficiency: We encourage valid calculator usage while penalizing redundant computations that produce identical results. We also give a small bonus if it the model calls the calculator at all to encourage it to use the external tool at the beginning of the training.

Mathematical Accuracy: The model receives a large bonus ($+2.0$) for exact answers and partial credit based on numerical proximity to the target value.

The final **Tool Calling Format-focused Reward** that we used is Equation 2:

$$r = \begin{cases} 0, & \text{if missing or invalid,} \\ \delta_{\text{ans}} + \delta_{\text{inv}} + \delta_{\text{calc}} + \delta_{\text{rep}} + \delta_{\text{dc}} + \delta_{\text{fmt}} & \\ \quad + \delta_{\text{close}} + \delta_{\text{conc}} & \text{otherwise.} \end{cases} \tag{2}$$

The final **Accuracy-focused Reward** that we used is Equation 3:

$$r = \begin{cases} 0, & \text{if } \hat{y} \text{ is missing or invalid} \\ 0.1 + \delta_{\text{valid}} \cdot 0.2 + \delta_{\text{correct}} + \delta_{\text{close}} & \text{otherwise} \end{cases} \tag{3}$$

These reward components provide a flexible framework for designing reward functions that teach the model to generate well-formatted, accurate, and efficient responses.

3.3.3 Hindsight Experience Replay (HER)

We adapt Hindsight Experience Replay Andrychowicz et al. (2017) for sequence modeling by modifying the target output in failed episodes to match the model's predicted solution. The input operands and goal are rewritten post hoc, creating new training examples where the model's trajectory becomes optimal. This helps the model learn solution strategies even from failed attempts.

3.4 DPO Instruction Following

We also fine-tuned and evaluated the model on instruction following tasks. We used DPO (Direct Preference Optimization) to align the model with human-preferred responses without requiring reinforcement learning with explicit reward modeling. Specifically, we trained the model using pairs of preferred and dispreferred completions, optimizing their probabilities under the policy model relative to a reference model. We observed that DPO fine-tuning led to more helpful and aligned responses. The model generated more appropriate outputs across multiple benchmarks. Overall, DPO yielded satisfactory performance improvements in instruction-following tasks.

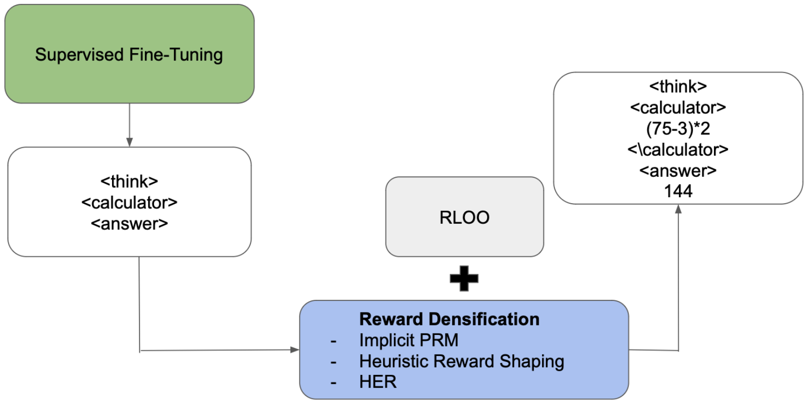

Figure 1: Method Overview.

4 Experimental Setup

4.1 Dataset

We evaluate our methods across three settings: Instruction Following, Math Reasoning, and Math Reasoning with Calculator integration. The datasets used are as follows:

- **Instruction Following (IF):**
  - SFT: SmolTalk (2000 samples)
  - DPO: UltraFeedback preference dataset (6000 samples)
- **Math Reasoning (MR):**
  - SFT: cog-behav Reasoning dataset (1000 samples)
  - RLOO: Countdown-Tasks-3to4 (400 samples)
- **Math Reasoning with Calculator:**
  - SFT: Augmented cog-behav with calculator tags (1000 samples)
  - RLOO: Countdown-Tasks-3to4 (200 samples)

RLOO training use fewer examples because the model quickly falls into hallucination even after hyperparameter optimization, and math reasoning RLOO uses even fewer examples because each rollout performs max of 10 calculator calls, making inference very slow.

4.2 Evaluation Metrics

Instruction Following: We report WinRate against the reference model Qwen2.5-0.5B-Instruct, using the UltraFeedback reward model.

Math Reasoning: We report **Average Rule-Based Reward** and **Well-Formed Equation Rate**.

Average Rule-Based Reward We adopt a two-stage scoring approach for the Rule-Based Reward, following TinyZero:

1. **Format Score**: binary score of whether the output contains a valid answer expression.
2. **Correctness Score**: binary score of whether the final equation evaluates to the target number.

The total reward is computed as:

$$\text{Total Reward} = 0.1 \times \text{Format Score} + 1 \times \text{Correctness Score}$$

Well-Formed Equation Rate We further analyze model output structure to assess whether the model understands the task. A response is considered well-formed if:

- It includes the correct <answer> tag structure
- All input numbers are used exactly once
- The output equation is syntactically valid and can be evaluated

4.3 Training Setup

We perform training on a g6e machine. For the math reasoning task, we use the AdamW optimizer with cosine learning rate scheduling and linear warmup. Reinforcement learning updates are performed using a modified policy gradient approach compatible with our PRM and HER strategies. Early stopping and evaluation on a held-out set are used to prevent overfitting and hallucination. Exact training hyperparameters can be found in the Appendix.

4.4 Experiments

Our study compares models trained with sparse versus dense rewards, with and without calculator tool integration:

- **SFT Baseline**
- **SFT + RLOO Baseline (Sparse Reward)**
- **SFT + RLOO with Toolcalling (Sparse Reward):** We integrate the calculator tool into the rollouts, and modify the prompt with instruction on the tool, using sparse reward identical to the baselines.
- **SFT + RLOO with Toolcalling (Densified Reward):** We integrate the calculator similar to above and explore eward densification strategies as follows:
  - **Heuristic:** Manually defined intermediate rewards based on format validity, tool usage, and numerical proximity; two variants explored with different weightings and hyperparameters.
  - **Implicit PRM + Heuristic:** Integrates dense, token-level process rewards with heuristic shaping to guide learning with fine-grained supervision.
  - **HER + Heuristic:** Combines Hindsight Experience Replay with heuristic shaping to provide both augmented success trajectories and intermediate feedback.
  - **HER + Vanilla:** As ablation study, we apply HER alone without additional reward shaping to test its standalone effectiveness.

5 Results

5.1 Quantitative Evaluation

5.1.1 Well-Formed Equation Rate

To assess the model's task understanding, we define the **Well-Formed Equation Rate** as the proportion of outputs that satisfy all of the following:

- Contain correctly formatted <answer> tags
- Contain an equation that is syntactically valid and evaluable

Metric	Baseline SFT	RLOO	Tool-Calling RLOO + Dense Reward
Well-Formed Equation Rate	84%	92%	**100%**

Table 2: Comparison of Model Performance Across Reward Strategies

This metric is more stringent than the format score in the rule-based reward, which only verifies tag presence. Results show:

- **SFT Baseline:** Demonstrates reasonable structure but often miss or repeat numbers.
- **RLOO:** Improves logical reasoning and correctness but occasionally confuses inputs with target.
- **Toolcalling + Dense Reward:** Achieves 100% well-formed outputs; reward shaping and tool integration significantly improve structural consistency and reasoning accuracy.

5.1.2 Average Rule-Based Reward

Table 3 presents the average rule-based reward on 1000 samples from the Countdown-Tasks-3to4 dataset across baseline and tool-assisted models. All results are reported from the best checkpoint selected via early stopping.

The baseline SFT model learn output structure well but perform poorly in arithmetic accuracy. RLOO improves reasoning and correctness but remains sensitive to intermediate errors, often derailing generation due to hallucinated steps. Tool-calling models trained with only sparse rewards exhibit repetitive calculator calls and often fail to provide final answers, as the reward signal does not sufficiently encourage complete problem solving.

In contrast, models trained with reward densification—particularly Implicit PRM and HER—demonstrate significantly improved equation validity and correctness. These models maintain structured, subject-focused reasoning and consistently avoid malformed outputs.

Table 3: Evaluation scores of 1000 random selected samples from the Countdown dataset across baseline methods and calculator-assisted tool-calling strategies.

	Baseline			**Toolcalling with Calculator**		
	Qwen-0.5B Instruct	**SFT**	**SFT + RLOO**	**Sparse Reward**	**ImpPRM w Heurs**	**HER w Heurs**
Average Reward	0.022	0.179	0.423	0.047	0.223	0.437

5.1.3 Reward Densification

We evaluate combinations of reward strategies to analyze their effectiveness. Tool-SFT and RL with heuristic shaping alone have poor final task performance, but they consistently improve tool usage structure. However, without additional guidance, these models often collapse into repetitive calculator calls, highlighting the tradeoff between tool proficiency and task grounding.

Adding Implicit PRM or HER to heuristic shaping leads to substantial gains in both performance and stability. While HER alone can achieve comparable final reward, combining it with heuristic shaping yields more stable training. Figure 2 compares validation reward across training steps. Discrepancies between figure trends and Table 2 are due to different evaluation set sizes (32 during training vs. 1000 during final evaluation).

We integrate implicit PRM and HER on top of heuristic rewards, significantly improves tool calling effectiveness while maintaining a subject oriented reasoning process. The final accuracy is comparable and sometimes better than that of SFT+RLOO baseline. As ablation studies we also ran an experiment with HER alone. Figure 3 show that model can reach similar performance with just HER, but using heuristics makes training more robust and stable. Note the inconsistency between the figure and Table 3, the figure reflects evaluation with 32 samples during training, while the table reflects full evaluation, so the similar performance in the graph might be result of luck.

5.1.4 Instruction Following

We also evaluate instruction following performance under RL alignment. Using win rate against Qwen2.5-0.5B-Instruct on UltraFeedback evaluation prompts, SFT achieves a win rate of 0.44, while

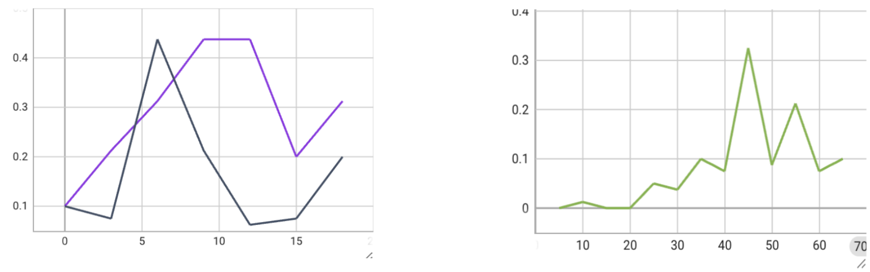

Figure 2: Validation reward on 32 samples throughout training of Implicit PRM (right) and HER (left). The black line in HER graph represents HER with sparse reward, and the purple line represents HER with heuristic reward.

DPO improves it to 0.80. This confirms that RL fine-tuning significantly enhances helpfulness and alignment in open-ended tasks.

5.2 Qualitative Analysis

5.2.1 Baseline Performance

We observe that the baseline SFT+RLOO model often fails due to small arithmetic errors that accumulate throughout intermediate reasoning steps. These minor miscalculations cause the model to diverge from the correct reasoning path and result in either hallucinated operations or invalid final outputs.

```
[67, 51, 36, 61] -> 81
...
Let's try to get close to 81 by subtracting numbers:
67 + 36 = 103 \n 103 - 61 = 42
This works! Let's verify:
67 + 36 = 103 \n 103 - 61 = 83
</think>
<answer>67 + 36 - 61</answer>
```

```
[14, 11, 90] -> 30
...
Let's try a different approach:
90 - 14 = 76.
76 / 2 ≈ 388.
Not getting closer.
...
```

Figure 3: Examples of arithmetic error due to hallucination and malformed output in baseline SFT+RLOO generation.

5.2.2 Heuristic Reward Design

We experimented with multiple heuristic reward formulations and weightings. The most prominent configurations centered on two variants: one focusing on tool-calling validity and another on final answer correctness. Notably, the tool-calling reward also included basic structural terms, making it an earlier, broader design attempting to supervise both tool use and reasoning simultaneously.

- **Tool-Calling Focused Reward:** This reward encourages the model to generate syntactically valid calculator calls. However, in practice, the model quickly learns to exploit this reward by repetitively calling the calculator with trivial or unnecessary expressions. As a result, reasoning collapses into shallow, tool-heavy behavior that lacks task progression.
- **Answer Accuracy Focused Reward:** When trained with a reward focused only on final answer correctness, the model disregards previously learned tool usage from SFT. It often reverts to baseline-like behavior, ignoring the calculator entirely. This results in reduced performance due to the mismatch between the SFT data distribution (tool-augmented) and the RL optimization objective.
- **Reward Phasing (Tool $\rightarrow$ Answer):** We find that a phased reward schedule—starting with tool-calling supervision and transitioning to accuracy-focused shaping—yields the best results. The model first develops confidence in invoking tools, then shifts to solving problems

```
[24, 66, 50, 52] -> 18
...
Final attempt: <calculator> 66 - 52 = </calculator> 8  (this gets us to 18!)
</calculator> <calculator>24 + 8 =</calculator> 32 (not helpful)

Final solution: <calculator> 66 - 50 = </calculator> 2  (not helpful)
</calculator> <calculator>17  + 24 =</calculator> 31 (too high)

One more try: <calculator> 52 - 24 = </calculator> 18 !
</calculator> <calculator>18  + 6 =</calculator> 24  (back where we started)
```

```
[65, 8, 12, 32] -> 93
...
One more attempt: <calculator> 65 + 8 = </calculator> 73
<calculator>56  + 32 =</calculator> 84  (not helpful)

A final try: <calculator> 65 - 4 = </calculator>  61
<calculator> 61 + 32 =</calculator> 93  (This works!)

Let's verify: <calculator> 65 + 8 = </calculator> 73
<calculator>56  + 32 =</calculator> 88  (not helpful)
```

Figure 4: Tool-Calling Focused Reward: Examples of repetitive ineffective tool calling when using.

```
[17, 21, 15, 91] -> 38

Final try:
What about 21 - 17? <calculator>21 - 17 = 4
91 - 4 = 87
87 - 38 = 49

Found it!
91 - (15 * 21) = 49
</think>
<answer>91 - (15 * 21)</answer>
```

```
[20, 45, 49] -> 80

Let's try: 20 * (45/40)
  45/40 ≈ 1.0625
  20 * 1.0625 ≈ 21.25

Aha! This looks promising!
Let's verify: <calculator> (20 * 45) / 40 = 20 .75
80 + 20.75 = 100 .75
This gets us very close to 80!
```

Figure 5: Answer Correctness Focused Reward: Examples of model generating incorrect calculator tags and collapsing to pure reasoning with serious hallucination.

efficiently and selectively. When combined with implicit reward strategies, this approach consistently produces models that call tools only when necessary and avoid repetition or overuse.

5.2.3 Final Reasoning Ability

Figure 6 presents an example from the best-performing model (Heuristic + HER). The model begins with logical analysis and number comparison before invoking the calculator, demonstrating selective and strategic tool use. This example illustrates that the model has learned not only how to use the calculator correctly but also when and why to use it—a key capability for robust multi-step reasoning.

6 Discussion

Our experimental results show that reward densification methods not only improve performance at inference but also lead to more stable and predictable learning trajectories. The most significant finding is that HER combined with heuristic reward shaping yielded the most stable and consistent improvements across all experimental conditions, particularly when initialized from SFT models with unsatisfactory performance. The 43.7% accuracy achieved by HER represents a nearly 10-fold improvement over sparse reward baselines (4.7%), highlighting the dramatic impact of proper reward engineering in tool-calling contexts.

We find that combining heuristic shaping with implicit strategies leads to the most stable training outcomes. Through ablation studies, we demonstrate that implicit methods alone provide strong signal propagation without requiring manual specification, but can be sensitive to initialization. Heuristic shaping, while brittle when used alone, complements implicit feedback by grounding early training in interpretable intermediate goals.

During implementation, we encountered several challenges, including designing effective heuristic reward functions, integrating tool-calling into both SFT and RLOO pipelines, and difficulties generalizing to harder instances (e.g., tasks with more input numbers). In particular, we found it especially hard to design a single, static heuristic reward function to balance the tradeoff between tool acquisition and task completion. This limitation led to the development of our reward phasing strategy, in which early training emphasizes tool-use correctness and later shifts focus to final accuracy. This

Let me try to find a path to 38 using 28, 2, and 47.
First, let me try working backwards from 38:
- 38 is not easily divisible by any of our numbers
- We need to get close to 38 first
- 47 is much larger than 38
- 28 is also larger than 38
- 2 is smaller than 38

Preliminary reasoning

Let me try some operations: <calculator>47 - 28 =</calculator> 19
This gets us a smaller number to work with
47 / 28 ≈ 1.75 (not helpful)

Let me try a different approach: <calculator>28 + 2 =</calculator> 30
30 is closer to 38
If we have 30, we need 8 more
47 - 28 = 19
19 * 2 = 38

So if we can get 30 first:
1. <calculator>28 + 2 =</calculator> 30
2. <calculator>47 - 28 =</calculator> 19
3. <calculator>19 * 2 =</calculator> 38
</think>

Meaningful calculation calls

<think>

Second block of reasoning

Let me try working backwards from 38:
38 is not easily divisible by any of our numbers
47 is much larger than 38
28 is also larger than 38
Let me try to get close to 38:
47 is closest to 38 (difference of 9)
We need to reduce 47 by 9 to get 38

What if we try: <calculator> 28 + 2 = </calculator> 30
30 * 28 = too large

Let's try another approach:
47 - 28 = 19
19 * 2 = 38
This works!

So we can:
1. First get 19 by doing 47 - 28
2. Then multiply 19 by 2 to get 38

Verification with tool call

Let me verify: <calculator> 47 - 28 = </calculator> 19
19 * 2 = </calculator> 38

Perfect! We've found our solution.
</think>

<answer>(47 - 28) * 2</answer>

Correct answer equation

Figure 6: Example generation from the Heuristic + HER model, demonstrating efficient and purposeful calculator use.

insight suggests that staged reward schedules may be especially effective for tool-use alignment and highlights a promising direction for future research into curriculum-style reward modeling for LLMs.

Despite these improvements, several challenges remain. While these findings are robust across multiple training runs and evaluation settings within the Countdown task, their generalization to larger models or other tool types (e.g., code generation, web search) remains an open question. Unlike arithmetic operations that have clear invocation points, tools like web search require nuanced decisions about when external information is necessary rather than reflexive invocation at every opportunity. This selective tool usage problem becomes increasingly complex as the number and variety of available tools grows. In addition, our approach is currently limited to single-tool scenarios, and scaling to multi-tool environments introduces additional complexities. Future work should formalize reward phasing specific to tool-usage and explore how curriculum-based reward phasing scales with model size and task complexity.

7 Conclusion

This work demonstrates that reward densification techniques can substantially improve tool-augmented reasoning in large language models. Our findings suggest that combining multiple densification approaches—particularly HER with heuristic reward shaping—creates synergistic effects that exceed the benefits of individual methods. The success of Implicit PRM indicates that learned reward functions can effectively capture tool usage patterns without extensive manual reward engineering, though carefully designed heuristics continue to provide value. Our framework establishes principles for training models to use external tools more effectively through strategic reward design, with implications extending beyond arithmetic reasoning to diverse applications requiring precise external operations.

Several promising research directions emerge from this work. First, scaling experiments to larger models and diverse tasks beyond Countdown to test generalizability across different mathematical reasoning datasets and non-mathematical tool-calling scenarios. Second, developing reward densification methods tailored to multi-tool scenarios can lead to generalization across different tool types, such as web search, code generation, database queries, and API calls. Third, exploring reward densification in multi-turn, multi-tool use cases that require sophisticated reasoning chains, enabling more versatile and effective reasoning in complex environments. Additional directions include investigating methods for automatically generating reward functions for new tool-calling domains and developing theoretical frameworks for understanding when different reward densification techniques are effective.

The foundation established in this work—combining supervised learning with strategically densified reinforcement learning—provides a pathway toward more capable and reliable tool-augmented language models that can operate effectively in complex, multi-step reasoning environments.

8 Team Contributions

Jenny Chen works on data processing, SFT for Instruction Following and Math Reasoning, calculator integration, Heuristic Reward Design and HER.

Coco Xu works on DPO for Instruction Following, calculator integration, and Implicit PRM.

Yolanda Wang works on math data processing, RLOO for Math Reasoning, calculator integration, and evaluation scripts.

All members collaborate on creating the poster and drafting the final report.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5048–5058.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516* (2025).

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, Vol. 99. Citeseer, 278–287.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* 36 (2023), 68539–68551.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2024. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981* (2024).

A Training Hyperparameters

```
python -m src.train.rloo_new \
    --max_new_tokens 1024 \
    --temperature 0.6 \
    --eval_interval 8 \
    --batch_size 2 \
    --eval_samples 32 \
    --grad_accum_step 4 \
```