🪜 Adaptive Temperature Ladder Sampling (ATLaS) for REINFORCE Leave-One-Out (RLOO)

A default project for CS 224R: Deep Reinforcement Learning[†]

Jonathan Algar
Non-Degree Student, Stanford Center for Global & Online Education
jalgar@stanford.edu

Abstract

REINFORCE Leave-One-Out (RLOO) offers an efficient alternative to Proximal policy optimization (PPO) for Large Language Model (LLM) RL-based fine-tuning but suffers from sparse rewards in mathematical reasoning tasks where correct solutions are rare. We introduce **Adaptive-Temperature Ladder Sampling (ATLaS)**, a simple yet effective extension that systematically increases sampling temperature when correct solutions are not found, recovering training signals from previously intractable problems. Our approach uses temperature as an exploration mechanism while maintaining sample quality. On our Countdown benchmark, ATLaS improves the score by a relative 8.4% over vanilla RLOO by successfully solving 11.4% of problems that generate zero correct solutions under fixed sampling. This work demonstrates that addressing the exploration-exploitation trade-off through adaptive temperature control can enhance RL fine-tuning for challenging reasoning tasks.

1 Introduction

LLMs demonstrate remarkable capabilities across diverse tasks, yet continue to struggle with multi-step numerical reasoning problems. The Countdown task exemplifies this challenge: given a small set of integers (for example, {4, 31, 35, 62}), the task requires combining them with basic arithmetic operations to exactly reach a target integer (for example, 41). Despite its apparent simplicity, both base and instruction-tuned versions of `Qwen2.5-0.5B` achieve zero accuracy our Countdown benchmark without specialized fine-tuning.

Reinforcement learning from human feedback (RLHF) has become an important technique for improving LLM capabilities beyond supervised fine-tuning Ouyang et al. (2022). Among RL algorithms for LLM fine-tuning, RLOO Ahmadian et al. (2024) offers an attractive balance of simplicity and effectiveness, requiring 50-70% less memory than PPO while achieving competitive performance. However, RLOO faces a significant limitation when applied to tasks with sparse rewards. Without sufficient correct samples to estimate meaningful advantage baselines, the algorithm cannot differentiate between attempts of varying quality.

In this work, we propose Adaptive-Temperature Ladder Sampling (ATLaS), a simple modification to RLOO that addresses the sparse reward challenge through intelligent exploration. The key insight is that temperature control in sampling can serve as an effective exploration mechanism. While low temperatures preserve high-quality samples when the model performs well, strategically increasing temperature when correct solutions are absent can reveal previously hidden solution paths.

[†] *Updated since submission. Last update: June 18, 2025.*

2 Related work

2.1 Supervised Fine-Tuning (SFT)

Supervised fine-tuning forms the foundation for subsequent RL fine-tuning methods. The standard RLHF pipeline begins with SFT on high-quality demonstrations Ouyang et al. (2022), which proves particularly important for mathematical reasoning tasks. Recent work by Dong et al. (2023) shows that mathematical capabilities improve consistently with increasing data quantity, though general abilities plateau after approximately 1000 examples. This highlights the importance of specialized datasets for mathematical tasks.

2.2 REINFORCE Leave-One-Out (RLOO)

RLOO represents a significant advancement in policy gradient methods for LLM fine-tuning. Initially proposed by Kool et al. (2019) for combinatorial optimization, RLOO was adapted for LLM fine-tuning by Ahmadian et al. (2024). The key innovation lies in its variance reduction technique: for each sample in a batch, RLOO uses the average reward of all other samples as a baseline.

The theoretical foundation of RLOO builds on the insight that multiple samples from the current policy can serve dual purpose: contributing to gradient estimation while simultaneously providing baselines for variance reduction. Recent implementations have shown that RLOO achieves performance comparable to PPO while using 50-70% less vRAM and running 2-3x faster Huang and Ahmadian (2024).

2.3 Temperature control in LLMs

Temperature sampling has long been recognized as important for controlling LLM output diversity Graves (2013). In the softmax function, temperature τ scales the logits: $p_i = \exp(z_i/\tau)/\sum_j \exp(z_j/\tau)$. Lower temperatures concentrate probability mass on high-likelihood tokens, while higher temperatures increase diversity.

Recent work explores adaptive temperature control for various applications. Zhu et al. (2024) propose learning token-wise temperature controllers for code generation, while Renze (2024) investigate temperature scheduling for improved sample quality. However, these approaches typically focus on inference-time generation rather than train-time exploration. Our work differs by coupling temperature adaptation with RL's reward signal, using it as an exploration mechanism during policy optimization.

2.4 Addressing sparse rewards in RL

The sparse reward problem is well-documented in RL literature. Traditional approaches include reward shaping Ng et al. (1999), curriculum learning Bengio et al. (2009), and exploration bonuses Bellemare et al. (2016).

For LLMs specifically, Lightman et al. (2023) introduce process supervision, providing intermediate rewards during mathematical reasoning. However, this requires expensive step-by-step annotations. Self-rewarding mechanisms Chen et al. (2024) offer an alternative but rely on fixed heuristics that may not generalize across problem types.

Our approach requires no additional annotations or reward engineering while maintaining the simplicity of outcome-based rewards.

3 Method

3.1 RLOO foundation

RLOO operates by generating K samples $\{y_1, ..., y_K\}$ for each prompt x and computing advantages using leave-one-out baselines:

$$b(x, y_k) = \frac{1}{K-1} \sum_{i=1, i \neq k}^{K} R(x, y_i) \tag{1}$$

The policy gradient update follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\frac{1}{K} \sum_{k=1}^{K} A(x, y_k) \nabla_\theta \log \pi_\theta(y_k|x) \right] \tag{2}$$

This formulation provides unbiased gradient estimates while leveraging all samples for variance reduction.

3.2 The ATLaS algorithm

ATLaS extends RLOO by introducing adaptive temperature control during sampling. The key insight is that when initial sampling at base temperature τ_0 fails to produce correct solutions, incrementally increasing temperature can reveal solution paths while maintaining overall sample quality.

Algorithm 1 Adaptive-Temperature Ladder Sampling (ATLaS)

```
Require: Prompts $\{x_1, \ldots, x_B\}$; base temperature $\tau_0$; increment $\Delta\tau$; max attempts $M$; min correct samples $C_{min}$
Ensure: Samples and advantages for each prompt
for each prompt $x_i$ do
    $\mathcal{S}_i \leftarrow \emptyset$                                  ▷ Initialize sample set
    for $m = 0$ to $M - 1$ do
        $\tau_m \leftarrow \tau_0 + m \cdot \Delta\tau$
        Generate $K$ samples $\{y_1^{(m)}, ..., y_K^{(m)}\} \sim \pi_\theta(\cdot|x_i, \tau_m)$
        Compute rewards $\{r_1^{(m)}, ..., r_K^{(m)}\}$
        $C_i^{(m)} \leftarrow |\{j : r_j^{(m)} = 1.0\}|$                  ▷ Count correct samples
        if $C_i^{(m)} \geq C_{min}$ or $m = M - 1$ then
            $\mathcal{S}_i \leftarrow \{y_1^{(m)}, ..., y_K^{(m)}\}$
            break
        end if
    end for
    Compute RLOO advantages for $\mathcal{S}_i$ per Eq 2
end for
```

3.2.1 Design choices

Several design choices in ATLaS merit explanation:

Linear temperature schedule: We use a simple linear increase ($\tau_m = \tau_0 + m\Delta\tau$) rather than exponential or learned schedules. This provides predictable exploration increases while maintaining interpretability.

Prompt level adaptation: Temperature adaptation occurs per prompt rather than per batch or globally. This allows the algorithm to adapt to varying problem difficulties within a single batch.

Early stopping: Once sufficient correct samples are found, we stop increasing the temperature. This preserves sample quality for problems the model can already solve.

Fixed sample size: We maintain K samples regardless of temperature, ensuring consistent advantage estimation across all prompts.

3.2.2 ATLaS Integration with RLOO

Our implementation modifies the standard RLOO training loop to incorporate ATLaS while maintaining computational efficiency. The key optimization is selective resampling:

```python
for idx, (samples, rewards) in enumerate(zip(all_samples, all_rewards)):
    correct_count = sum(1 for r in rewards if r == 1.0)

    if len(samples) < self.config.num_samples:
        needs_more = True
    elif correct_count < self.config.min_correct_per_prompt:
        needs_more = True
    else:
        needs_more = False

    if needs_more:
        prompts_to_generate.append(prompts[idx])
        prompt_mapping[len(prompts_to_generate) - 1] = idx
```

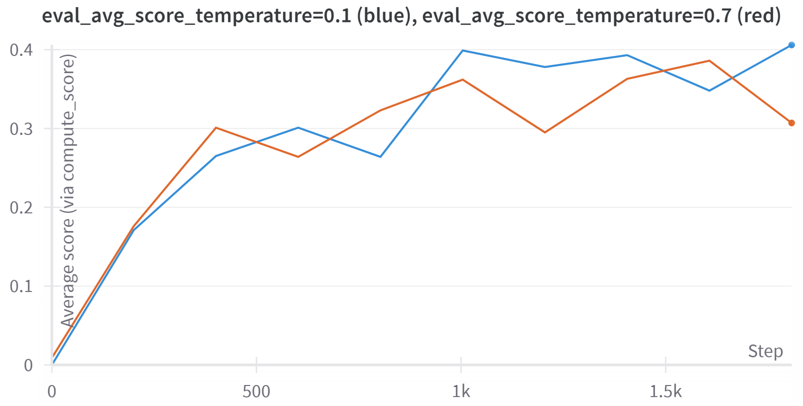
See Appendix A for example of log output from ATLaS.

4 Experimental setup

4.1 SFT baseline

We fine-tune `Qwen/Qwen2.5-0.5B` on the `cog_behav_all_strategies` dataset (train split, `seed=42` for reproducibility), which contains 1000 high-quality examples of mathematical reasoning with explicit backtracking and verification strategies. The hyperparameters used for training:

- Learning rate $\alpha = 1e^{-5}$
- Batch size = 2
- Maximum sequence length = 1024 tokens
- Training epochs = 4



(Click for interactive data)

4.1.1 Our Countdown benchmark

We evaluate using the `compute_score` function from Gandhi et al. (2025) over 100 samples from a synthetic validation set. We then select the checkpoint with best average score for `temperature=0.1` (=step 1800) as π_θ for RLOO training. For full training detail and experiment logs, see the W&B run overview.

Important note: Our SFT baseline achieves a 0.406 score on our Countdown benchmark, which is remarkably strong compared to the base model. This means we are not operating in a truly sparse

CS224R Methodological note: half the examples from the *original* test set which was replaced as leaderboard evaluator before we trained this model i.e. we made an `Instruction Following` submission to the project milestone leaderboard.

reward regime, as well over one-third of generated samples receive positive rewards even before RL fine-tuning. This strong starting point may underestimate the benefits of ATLaS compared to scenarios with genuinely sparse rewards.

4.2 RLOO training

For our RLOO training procedure we construct prompts (template as `cog_behav_all_strategies`) from the `Countdown-Tasks-3to4` binary dataset (train split, `seed=42` for reproducibility). For each gradient step we sample $K = 20$ candidate completions per prompt with a batch size of 8 (selected for computational feasibility). We make a fixed 101 gradient steps for the run. We use a AdamW optimizer with learning rate $\alpha = 1e^{-5}$ and KL penalty coefficient $\beta = 0$.

These are the common hyperparameters and configuration for our RLOO train runs. We want to evaluate vanilla RLOO versus RLOO+ATLaS. The configuration for the ATLaS algorithm in our experimental setup:

- Base temperature $\tau_0 = 0.7$
- Temperature increment $\Delta\tau = 0.1$
- Maximum attempts $M = 3$ ($\Rightarrow$ maximum temperature $\tau_{\max} = 0.9$)
- Minimum correct samples required $C_{min} = 1$

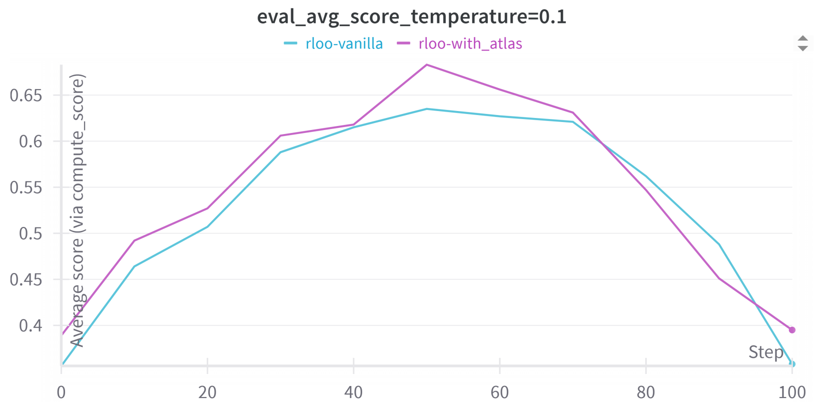
For vanilla RLOO we set $M = 1$.

The headline metric will be model performance on the Countdown benchmark measured at a 20-step interval using the same protocol as outlined for our Countdown benchmark outlined above ("Best average score").

5 Results

The table below presents our headline experimental result as aggregated over two independent runs (1x Vanilla RLOO, 1x RLOO+ATLaS).

Method	Best average score	Relative train time
Vanilla RLOO	0.647	1.0x
RLOO+ATLaS	**0.683**	1.57x



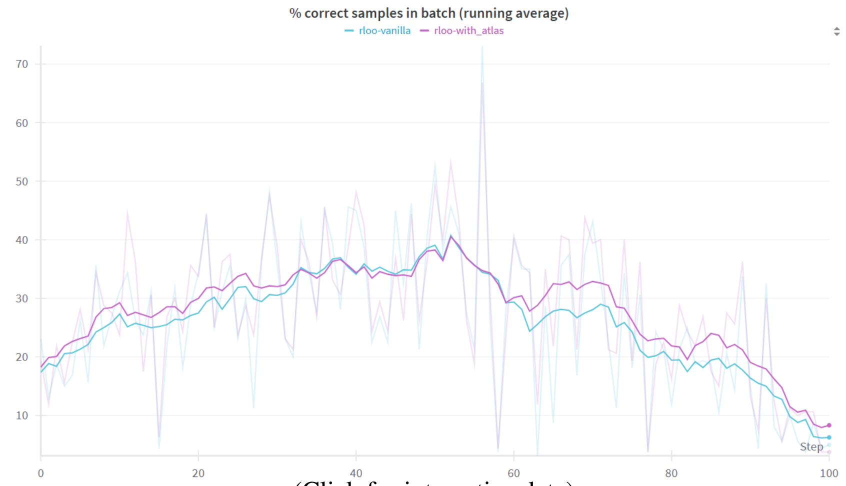
(Click for interactive data)

ATLaS achieves a modest **8.4% relative improvement in best average score over vanilla RLOO.** The computational cost is significant, with a 1.57x increase in train time.

Important note: Due to computational budget constraints, we conducted only one train run for each method. This prevents us from establishing statistical significance or confidence intervals for the reported improvements. Then the observed relative improvement, while encouraging, should be interpreted as a preliminary finding requiring validation through multiple independent runs.

5.1 Resampling

For the **RLOO+ATLaS** run we find that of the 808 prompt (8 x 101 steps) sample sets analyzed, 306 (37.9%) did not have correct solution after the first attempt. Of these, 35 (11.4%) were "saved by ATLaS" and had a correct solution on the 2nd ($\tau = 0.8$, 26 prompts) or 3rd ($\tau = 0.9$, 9 prompts) attempt.



(Click for interactive data)

6 Discussion

The temperature ladder mechanism can be understood through the lens of exploration bonuses in RL. Higher temperatures effectively implement a form of count-based exploration, encouraging the model to visit underexplored regions of the solution space. Unlike traditional exploration bonuses that require explicit state counting, temperature-based exploration emerges naturally from the sampling process.

There are several promising avenues for further research:

Statistical validation: The most immediate need is conducting multiple independent runs to establish statistical significance of the observed improvements and compute confidence intervals.

Truly sparse reward settings: Evaluating ATLaS on tasks where SFT baselines achieve near-zero performance would better demonstrate its benefits for genuinely sparse reward scenarios.

Theoretical analysis: Analysis of ATLaS's convergence properties and sample complexity could provide deeper insight into its effectiveness.

7 Conclusion

We introduce ATLaS, an adaptive temperature control mechanism that enhances RLOO's effectiveness on sparse-reward tasks. By systematically increasing exploration when needed, ATLaS recovers training signals from previously intractable problems, achieving an 8.4% relative improvement on our Countdown benchmark. This work demonstrates that addressing the exploration-exploitation trade-off through adaptive sampling can improve RL fine-tuning for challenging reasoning tasks.

7.1 Changes from proposal

Significant. We faced challenges with the base implementation of RLOO and its stability, so we decided to focus on an extension that does not require modifying the advantage calculation.

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).

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335* (2024).

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492* (2023).

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307* (2025).

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

Shengyi Costa Huang and Arash Ahmadian. 2024. *Putting RL back in RLHF*. `https://huggingface.co/blog/putting_rl_back_in_rlhf_with_rloo`

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! (2019).

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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.

Matthew Renze. 2024. The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 7346–7356.

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Zhi Jin, and Hong Mei. 2024. Hot or cold? adaptive temperature sampling for code generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 437–445.

A Example train run

Below is a snip of a train run log showing the ATLaS algorithm in action (full log here).
We see that on the first attempt only 5/8 prompts in the batch had correction solutions, but by after the third attempt 6/8 prompts have at least one correct solution.

```
...
Step 18/100

Processing batch of 8 prompt(s)
  Prompt 1: target=41 from [4, 31, 35, 62]
  Prompt 2: target=98 from [74, 28, 2, 72]
  Prompt 3: target=10 from [40, 94, 92, 5]
  Prompt 4: target=91 from [92, 91, 90]
  Prompt 5: target=39 from [66, 53, 18, 6]
  Prompt 6: target=45 from [32, 93, 16]
  Prompt 7: target=58 from [52, 7, 54, 12]
  Prompt 8: target=29 from [90, 30, 46, 15]
Using up to 3 attempts...
Generating 8 prompts x 20 samples...
After attempt 1: 37/160 correct (23.1%)
4 prompt(s) still lack correct solutions

Attempt 2/3 with temperature=0.80:
Regenerating for 4 prompts lacking correct solutions
After attempt 2: 38/160 correct (23.8%)
3 prompt(s) still lack correct solutions

Attempt 3/3 with temperature=0.90:
Regenerating for 3 prompts lacking correct solutions
After attempt 3: 39/160 correct (24.4%)

Generation phase complete...
Used 3 attempt(s), final temperature: 0.90
6/8 prompts have at least one correct solution

Sample selection summary:
  Prompt 1: Using samples from attempt 1 (temp=0.70) - 0 correct
  Prompt 2: Using samples from attempt 3 (temp=0.90) - 1 correct
  Prompt 3: Using samples from attempt 1 (temp=0.70) - 4 correct
  Prompt 4: Using samples from attempt 1 (temp=0.70) - 6 correct
  Prompt 5: Using samples from attempt 2 (temp=0.80) - 1 correct
  Prompt 6: Using samples from attempt 1 (temp=0.70) - 19 correct
  Prompt 7: Using samples from attempt 1 (temp=0.70) - 0 correct
  Prompt 8: Using samples from attempt 1 (temp=0.70) - 8 correct

Computing rewards for 8 x 20 samples...

Prompt 1 (target=41 from [4, 31, 35, 62]):
  Samples and scores:
   Sample 1: score=0.1, equation=’(35 + 31) - 62’
   Sample 2: score=0.1, equation=’((35 + 31) - 62) * 4’
   Sample 3: score=0.1, equation=’(62 - 35) + 31’
   Sample 4: score=0.1, equation=’(35 - 31) + 41’
   Sample 5: score=0.1, equation=’(62 - 31) + (35 - 31)’
   Sample 6: score=0.1, equation=’(31 + 35) - 62’
...
```