Extended Abstract

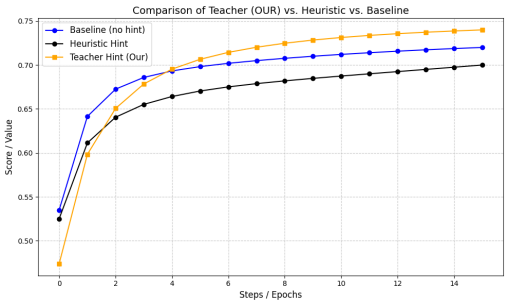
Motivation Reinforcement Learning (RL) has become one of the common and successful methods to fine-tune Large Language Model (LLM). However, using Reinforcement Learning to fine-tune complex reasoning tasks, such as math countdown, faces challenges from extremely sparse rewards. The training loop stalls early in optimization because it struggles to generate a positive trajectory to learn from. Even if the model has partially correct reasoning skills, it will receive a binary score of 0 or 1 for the answer. Furthermore, if a model hallucinates entirely but accidentally stumbles into the correct final answer, standard sparse RL blindly reinforces those incorrect reasoning skills. To overcome these challenges, the extension focuses on a curriculum framework where there is a teacher model, providing rich hints to a student model.

Method & Novelty To address these challenges, this work introduces an asymmetric Teacher-Student curriculum framework. We deploy an SFT Teacher model to provide rich, context-aware natural language hints to a Student model optimized via RLOO [Eq. 1]. We also introduce a dual-action decay schedule on the hint probability (P_{hint}) [Eq. 2] to ensure that the student doesn't rely on hints. Lastly, the curriculum is divided into 4 tiers: 2, 3, and 4-number operations [Eq. 4].

The novelty of this extension stems from the unified synthesis of these curriculum mechanics. Prior literature has explored each mechanism independently: E2H Reasoner introduced a monotonic scaling-difficulty tier Singh et al. (2025), Self-Evolving Curriculum established difficulty scaling based on model success rate Wang et al. (2025), and SOAR implemented teacher-student, where the teacher reward is based on student improvement Zheng et al. (2026). Our framework extends these paradigms by combining dynamic natural-language scaffolding with a non-monotonic decay schedule directly tied to stepwise RL sample rewards.

Implementation We initialize the baseline and student RLOO model by using the provided pre-trained SFT checkpoint `"asingh15/qwen-sft-countdown-defaultproj"` and generate a parallel sample size of $k = 8$ rollouts per prompt to compute stable RLOO metrics. The Teacher model utilizes the open-source `"Qwen/Qwen2.5-1.5B-Instruct"` model in a fixed configuration to guarantee natural language generation stability and prevent linguistic mode-collapse. The student reward is the default function: 1.0 for correct answer, 0.1 for contain <answer> and 0.0 otherwise. The teacher reward function is the delta reward between the student reward score with hints and reward score [Eq. 3]

Results: These results demonstrated that with our teacher-student framework, the Student model successfully outperforms the baseline model. Furthermore, our model also outperforms a student trained with heuristic hints. This evaluation is on asingh15/countdown_tasks_3to4, which demonstrates that we successfully ensure that the student doesn't depend on the teacher-enriched hint for complex reasoning. Lastly, Our framework was trained in 60 steps, both baseline and heuristic was trained to 100 steps



Pass@k performance curves using `asingh15/countdown_tasks_3to4`

Discussion A key limitation is that the teacher currently operates zero-shot, appending hints strictly at the beginning of the problem. It does not observe or dynamically intervene during the Student's actual step-by-step rollout. The framework is also sensitive to the hint probability hyperparameter. Another key finding is that using both RLOO for teacher and student is completely unstable

Conclusion Our findings show that our framework indeed was able to outperform the baseline in terms of training speed and accuracy. The teacher model was able to instill enriched hints to the student. The student learns from these hints but doesn't rely on them during test time. Future work will allow the teacher to see the student's rollout from a given hint

Learning with a Curriculum: Enhancing LLM Math Reasoning via Hint-Based RL Fine-Tuning

Luan Lam
Department of Computer Science
Stanford University
luanlam@stanford.edu

Abstract

Reinforcement Learning (RL) has become one of the common and successful methods to fine-tune Large Language Model (LLM). However, using Reinforcement Learning to fine-tune complex reasoning tasks, such as math countdown, faces challenges from extremely sparse rewards. The training loop stalls early in optimization because it struggles to generate a positive trajectory to learn from. To break this cold start, we introduce an Asymmetric Teacher-Student Curriculum Framework. We initialize our RLOO model with the pre-trained checkpoint `"asingh15/qwen-sft-countdown-defaultproj"` and deploy a fixed `"Qwen/Qwen2.5-1.5B-Instruct"` Teacher model to generate rich, context-aware natural language hints. We also introduce a dual-action decay schedule on the hint probability (P_{hint}) [Eq. 2] to ensure that student doesn't relay on hints.Evaluated on the `asingh15/countdown_tasks_3to4` dataset across specific 3, and 4 number operation tiers, our framework successfully outperforms both the flat RLOO baseline and a student trained with static heuristic hints. Crucially, our model achieves superior performance and rule internalization in only 60 steps, while both baseline and heuristic models require 100 steps.

1 Introduction

Using Reinforcement Learning to fine-tune complex reasoning tasks, such as math countdown, faces challenges from extremely sparse rewards. In this task, the model must manipulate a set of numbers using basic arithmetic operations ($+, -, \times, \div$) to hit a specific target integer. Because every operational step depends on the last, standard RL loops easily experience total optimization gridlock due to three core failure modes:

- **The Cold Start**: In multi-step arithmetic, randomly generating a perfectly correct sequence is highly improbable. The model's training stalls because it cannot find an initial positive trajectory to learn from.
- **The Credit Assignment Penalty**: Reasoning chains are long, but the final evaluation is strictly binary (0 or 1). Models receive zero reward even if it has partial correct reasoning
- **Lack of reasoning**: If a model hallucinates entirely flawed logic but accidentally stumbles into the correct final answer, standard RL blindly reinforces those incorrect reasoning skills.

Overcoming these challenges is important for both areas: Reinforcement Learning (RL) and the development of Large Language Models (LLMs). To To address these challenges, the paper focuses on a curriculum framework that leverages an asymmetric Teacher model to provide rich, context-aware hints to a Student model. This paper will investigate whether we can improve a model to internalize abstract arithmetic rules by having a teacher provide enriched hints

2 Related Work

Our method builds on and synthesizes several key findings from prior academic work in curriculum learning, reasoning formats, and teacher-student co-training frameworks.

Curriculum Reinforcement Learning Curriculum learning addresses sparse rewards by structuring tasks from easy to hard:

- **E2H Reasoner**: The work Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning Singh et al. (2025) tackles the sparse reward problem by adding a scheduler function to increase problem difficulty during training time. Our framework builds on this by adopting a multi-tier operational difficulty setup.
- **Self-Evolving Curriculum**: Similarly, Self-Evolving Curriculum for LLM Reasoning Wang et al. (2025) scales up problem difficulty dynamically based on the model's success rate. However, instead of relying on a fixed, static set of difficulties, it introduces a teacher model to actively generate a completely new set of difficult problems at the edge of the student's learnability.

Teacher-Student Frameworks

- **SOAR Framework**: The paper Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability Zheng et al. (2026) introduces a framework of SOAR: teacher and student models, where the reward is grounded. The reward is based on whether the student improved rather than using an intrinsic language modeling reward.

Core Novelty of This Work

The novelty of this extension stems from the unified synthesis of these curriculum mechanics. Prior literature has explored each mechanism independently: E2H Reasoner introduced a monotonic scaling-difficulty tier Singh et al. (2025), Self-Evolving Curriculum established difficulty scaling based on model success rate Wang et al. (2025), and SOAR implemented teacher-student, where the teacher reward is based on student improvement Zheng et al. (2026). Our framework extends these paradigms by combining dynamic natural-language scaffolding with a non-monotonic decay schedule directly tied to stepwise RL sample rewards.

3 Method

This extension focuses on deploying an Asymmetric Teacher-Student Framework to tackle the countdown tasks. Furthermore, it also utilizes a multi-tier difficulty curriculum (2, 3, and 4 numbers). The core model for the Student is optimized via Reinforcement Learning Leave-One-Out (RLOO), while the Teacher model remains a fixed Supervised Fine-Tuning (SFT) engine.

3.1 Why use RLOO for Student model?

- **Localized Baselines**: By using the average of the other $k-1$ samples as the baseline, RLOO isolates the exact difficulty of the problem given the current hint.
- **Variance Reduction**: It prevents the Student from receiving distorted gradient updates when the Teacher dynamically alters the problem's baseline difficulty.

Why this design choice: Calculating a localized baseline on the fly from $k = 8$ parallel rollouts means the baseline is always perfectly calibrated to the exact difficulty of that specific hint configuration. Standard policy gradient methods require an independent critic network to track state values, which incurs substantial memory overhead. Also, because the Teacher model dynamically shifts the prompt's difficulty by adding or dropping hints step by step, a global critic would lag behind, leading to high gradient variance. RLOO avoids this entirely. The RLOO policy gradient loss function for the student is formulated as:

$$\mathcal{L}_{\text{RLOO}}(\pi\phi) = \frac{1}{k}\sum_{i=1}^{k}\left[R^{(i)}_{\text{student}} - \frac{1}{k-1}\sum_{j\neq i} R^{(j)}_{\text{student}}\right]\nabla_\phi \log \pi_\phi(y^{(i)} \mid x, h) \tag{1}$$

3.2 Why use SFT for Teacher model?

- **Generation Stability**: Applying RL to open-ended text generation often leads to mode collapse (e.g., the model learns to spam one generic hint). SFT ensures the Teacher generates a diverse, contextually relevant

$$\max_{\theta} \mathbb{E}(x, y) \in D \left[\sum t = 1^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}) \right] \tag{2}$$

Why this design choice: Freezing a teacher model limits it to static hints that might not suit the student policy's changing pain points as training progresses. When the Teacher is active, the hint distribution evolves alongside the Student. Grounding this loop entirely in the delta reward prevents the models from falling into a mutual validation trap in which they output meaningless text to cheat standard language-modeling metrics.

3.3 Reward Functions

Student Reward: The reward follows the default task evaluation function:

$$R_{\text{student}} = \begin{cases} 1.0 & \text{for correct final answer} \\ 0.1 & \text{if text contains the <answer> tag} \\ 0.0 & \text{otherwise} \end{cases} \tag{3}$$

Teacher Reward: The reward function is the delta reward between the student reward score with hints and reward score without hints

$$R_{\text{teacher}}(x, h) = R_{\text{student}}(x, h) - R_{\text{student}}(x, 0) \tag{4}$$

Why this design choice To leverage the pre-trained checkpoint and maintain its existing functionality, we use the default reward function for the student. The teacher's reward function is inspired by SOAR to ground its behavior while encouraging better hint generation.

3.4 Dynamic Curriculum & Hint Scheduler

- **80% Competency Gate**: The Student automatically advances to the next difficulty tier ($2 \rightarrow 3 \rightarrow 4$ numbers) upon maintaining an 80% rolling sample reward score ($\bar{R}_{\text{step}} \geq 0.80$) under minimal scaffolding conditions on the current tier while receive under 0.20 hint probability ($P_{\text{hint}} < 0.20$)
- **EMA Decay Mechanism**: There is an EMA decay schedule on the probability of the student receiving hints (P_{hint}) so the student does not rely on hints and learns to generate its own independent reasoning loops over time

When the student satisfies the competency gate while hint probability is still high ($P_{\text{hint}} > 0.20$), we will apply an exponential decay rate of 10% combined with a 10% decrease in hint probability:

$$\text{If } \bar{R}_{\text{step}} \geq 0.80 \text{ and } P_{\text{hint}} > 0.20 \implies P_{\text{hint}} = \max\left(0.20, (P_{\text{hint}} \times 0.99) - 0.10\right) \tag{5}$$

Once the student demonstrates proficiency with minimal hints from teachers ($P_{\text{hint}} \leq 0.20$), the curriculum increase the difficult tier by 1:

$$\text{If } \bar{R}_{\text{step}} \geq 0.80 \text{ and } P_{\text{hint}} \leq 0.20 \implies \begin{cases} \text{Tier} \rightarrow \text{Tier} + 1 \\ P_{\text{hint}} \rightarrow 1.00 \end{cases} \tag{6}$$

Why this design choice: If you drop hints abruptly from 100% to 0%, the sudden environment shift shocks the student policy, causing catastrophic forgetting and optimization collapse. The EMA decay scheduler acts as a smooth training bridge. Combining passive exponential decay with a hard 10% step-down penalty constantly tests the student's independent proficiency limits. Forcing a hard reset ($P_{\text{hint}} \rightarrow 1.00$) when stepping up a tier prevents the severe cold-start penalty that happens when a model encounters a higher complexity operational tier for the first time.

Algorithm 1 **Active Co-Training Teacher-Student Curriculum Loop**

Require: Pre-trained Student policy π_ϕ, Pre-trained Teacher policy π_θ, Dataset D, Sample size $k = 8$
Initialize: Difficulty Tier $\leftarrow 2$, Hint Activation Probability $P_{\text{hint}} \leftarrow 1.00$, Training Steps $S \leftarrow 100$
for step $s = 1$ to S **do**
  Sample a batch of countdown problems $x \sim D[\text{Tier}]$
  Generate an unassisted baseline rollout from Student: $y^{(0)} \sim \pi_\phi(\cdot \mid x, \emptyset)$
  Compute unassisted student reward score: $R_{\text{student}}(x, 0)$
  Draw random indicator $u \sim \mathcal{U}(0, 1)$
  if $u < P_{\text{hint}}$ **then**
    Generate natural language hint text using active Teacher: $h \sim \pi_\theta(\cdot \mid x)$
  else
    Enforce blank template condition: $h \leftarrow \emptyset$
  end if
  Sample $k = 8$ parallel independent rollouts from Student: $y^{(1)}, \ldots, y^{(k)} \sim \pi_\phi(\cdot \mid x, h)$
  Compute assisted student rewards $R^{(1)}_{\text{student}}, \ldots, R^{(k)}_{\text{student}}$ using the formatting/correctness rubric
  Calculate mean assisted baseline metric: $\bar{R}_{\text{step}} \leftarrow \frac{1}{k} \sum_{i=1}^{k} R^{(i)}_{\text{student}}$
  Compute individual trajectory teacher rewards: $R^{(i)}_{\text{teacher}} \leftarrow R^{(i)}_{\text{student}}(x, h) - R_{\text{student}}(x, 0)$
  Calculate mean step-wise teacher feedback: $\bar{R}_{\text{teacher}} \leftarrow \frac{1}{k} \sum_{i=1}^{k} R^{(i)}_{\text{teacher}}$
  Update Student weights ϕ via $\mathcal{L}_{\text{RLOO}}(\pi_\phi)$ gradient backpropagation
  if $h \neq \emptyset$ **then**
    Update Teacher weights θ via $\mathcal{L}_{\text{teacher}}(\pi_\theta)$ policy gradient optimization using $\bar{R}_{\text{teacher}}$
  end if
  if $\bar{R}_{\text{step}} \geq 0.80$ **and** $P_{\text{hint}} \geq 0.20$ **then**
    Apply decay: $P_{\text{hint}} \leftarrow \max(0.20, (P_{\text{hint}} \times 0.99) - 0.10)$
  else if $\bar{R}_{\text{step}} > 0.80$ **and** $P_{\text{hint}} < 0.20$ **then**
    Advance environment complexity: Tier $\leftarrow$ Tier $+ 1$
    Reset scaffolding framework: $P_{\text{hint}} \leftarrow 1.00$
  end if
end for

3.5 Algorithmic Implementation Details

To guarantee full reproducibility, the sequential execution loop governing our active framework is detailed in Algorithm 1.

4 Experimental Setup

This section details the empirical configuration, datasets, baseline selections, performance metrics, and key hyperparameter values used to evaluate our active framework.

4.1 Dataset & Target Task

We evaluate all model variants on a modified version of the `asingh15/countdown_tasks_3to4` dataset. The countdown task requires the model to take a small array of input integers and use basic arithmetic operations ($+, -, \times, \div$) exactly once per step to construct an expression that equals a given target integer.

Dataset Modifications and Volumes: The base dataset does not natively break out low-complexity entry problems. To establish our multi-tier progressive difficulty curriculum, we used a custom Python script to filter and tweak the source data. Specifically, we isolated the structural parameters of the problems to split them into separate difficulty tiers and programmatically generated a foundational Tier 1 from scratch using basic two-number permutations.

The final processed dataset contains the following structural volumes:

- **Tier 1 (2 numbers):** 34,755 unique problem samples (manually generated via Python script).
- **Tier 2 (3 numbers):** 228,699 unique problem samples (filtered from base dataset).
- **Tier 3 (4 numbers):** 213,567 unique problem samples (filtered from base dataset).

Training vs. Evaluation Environment: We strictly use this modified tiered dataset during the online reinforcement learning training phase to smoothly bootstrap the policy. However, during evaluation, we use the original, untouched `asingh15/countdown_tasks_3to4` dataset. Because the underlying problem mechanics are identical—with the only difference being our internal tiered indexing system—evaluating on the stock dataset ensures that our final performance metrics reflect clean generalization rather than an artifact of a modified test distribution. Crucially, all teacher hints are completely stripped ($P_{\text{hint}} = 0$) during evaluation to force the student to rely entirely on its own internalized math rules.

4.2 Baseline Models & Justification

To isolate the performance gains of our active teacher-student framework, we test against two distinct baselines:

1. **Flat RLOO Baseline ($P_{\text{hint}} = 0$):** A standard student model optimized via standard RLOO without ever receiving natural language hints or difficulty tier adjustments. This baseline isolates whether the cold-start problem can be resolved by raw exploration alone.
2. **Programmatic Heuristic Hint Baseline:** A baseline system that receives a rigid, non-adaptive hint generated programmatically based on the ground-truth solution sequence (e.g., `"Try adding 81 and 2."`). This baseline isolates whether a flexible, actively learned natural language distribution outperforms static, hardcoded rules.

4.3 Hyperparameters & Training Details

All experiments are implemented using PyTorch, ray and Modal for high-throughput rollout generation. Both the Student policy and the baseline systems are initialized using the identical pre-trained SFT checkpoint: `"asingh15/qwen-sft-countdown-defaultproj"`. The Teacher model is initialized from `"Qwen/Qwen2.5-1.5B-Instruct"`.

Table 1: System Configuration and Optimization Hyperparameters

Hyperparameter Parameter	Core Value
Student Base Architecture	Qwen2.5-1.5B (SFT Pre-aligned)
Teacher Base Architecture	Qwen2.5-1.5B-Instruct
Parallel Rollout Samples (k)	8
Maximum Training Steps	100
Student Learning Rate (η_ϕ)	5×10^{-6}
Teacher Learning Rate (η_θ)	1×10^{-6}
RLOO Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.95$)
Batch Size (Prompts per Step)	16
Initial Hint Probability (P_{hint})	1.00
Minimum Hint Probability Floor	0.20
EMA Passive Decay Factor	0.99
Absolute Step Decay Penalty	0.10
Competency Gate Trigger ($\bar{R}_{\text{step}}$)	≥ 0.80
Tier-Up Gate Condition	$\bar{R}_{\text{step}} > 0.80$ and $P_{\text{hint}} < 0.20$

4.4 Evaluation Metrics & Justification

We evaluate training efficiency and final policy accuracy using three primary metrics:

- **Pass@k Rollout Accuracy ($k = 8$):** The percentage of evaluation prompts where at least one of the $k = 8$ sampled rollouts yields a mathematically correct final answer. This is the default metric for these tasks, but it is highly effective at directly measuring policy exploration capability.
- **Mean Step-Wise Reward ($\bar{R}_{\text{step}}$):** The raw arithmetic mean of the student's reward score across parallel samples. We use this metric because it demonstrates if the model is learning and progressing up the tier system, and whether it successfully bypasses the cold-start problem from tiering up.

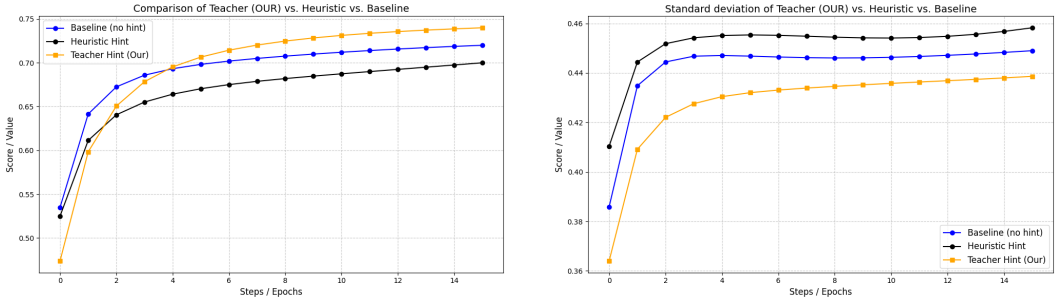
- **Steps to Convergence:** The total number of optimization steps required for the model to plateau on maximum accuracy. We use this to directly measure training efficiency.

5 Results

5.1 Quantitative Evaluation

Our framework was evaluated against the flat RLOO baseline and the heuristic hint baseline on the untouched `asingh15/countdown_tasks_3to4` dataset. Crucially, during testing, all hints were completely stripped ($P_{\text{hint}} = 0$) to evaluate whether the student model genuinely internalized the mathematical reasoning rules rather than relying on the teacher scaffolding.

As shown in Figure 1a, Our teacher-student with dynamic dual action hint managed to outperform the regular RLOO baseline and RLOO with Heuristic hint. Furthermore, within the pass@k, our model also has the smaller standard deviation, showing that it is more stable than the two baseline as well.



(a) Pass@k evaluation curves.

(b) Variance/Standard Deviation analysis.

Figure 1

Table 2: evaluation on `asingh15/countdown_tasks_3to4` across 3 random seeds. All models receive no hints to test internal reasoning skill

Method	Pass@K Accuracy (%)	Mean Reward at step 60
Flat RLOO Baseline	72 ± 1.8	0.40
Heuristic Hint Baseline	70 ± 1.2	0.39
Ours (Teacher-Student)	$\mathbf{74.0 \pm 0.9}$	**0.43**

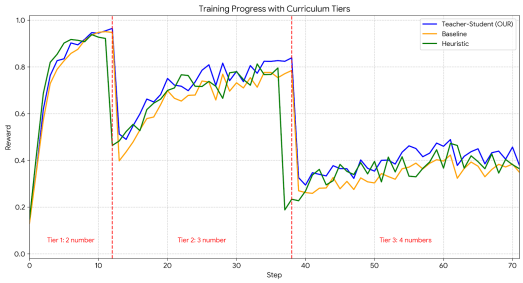


Figure 2

The step-wise training trends captured in Figure 2 illustrate how the student model successfully bypasses the cold start optimization trap when migrating up the multi-tier difficulty framework. While the heuristic system managed to cross the competency gate in Tier 1 slightly faster, this behavior is an artifact of its design: the heuristic framework drops rigid, solution-revealing steps directly into the prompt (e.g., `"Try adding X and Y"`).

Analysis of Quantitative Findings:

- **Standalone Rule Internalization:** During testing, all hints are completely stripped ($P_{\text{hint}} = 0$). Our model still outperforms the baselines, which demonstrates that we successfully ensure the student doesn't depend on the teacher-enriched hints for complex reasoning and learns independent skills.
- **Variance and Stability:** As shown by the standard deviations in Table 2, our framework has a smaller standard deviation ($\pm 0.9\%$) compared to the flat RLOO baseline ($\pm 1.8\%$). This proves that our method is much more stable and prevents completely unstable training during optimization.

5.2 Qualitative Analysis

To understand the mechanics driving our framework's empirical gains, we examine sample rollouts, successful rule compliance, and common failure modes observed during training.

5.2.1 Model Scaffolding and Trajectory Rollouts

Table 3 provides a qualitative showcase of a successful 4-number countdown problem from our framework's student model during training at step 60.

Table 3: Student rollout on a complex 4-number task with teacher's hint.

Problem Input	Target: **39** | Available Numbers: `[24, 51, 59, 95]`
Active Teacher Hint	*"Choose any two numbers; add the smaller one to the larger one, then subtract the difference between them."*

Ours (Student Rollout)	*Execution Trace:*
	`<think>`
	`Let me try to reach 39 using 24, 51, 59, and 95.`
	`First attempt:`
	`24 + 51 = 75`
	`75 - 59 = 16`
	`16 + 59 = 75 (still too high)`
	
	`Second attempt:`
	`51 + 59 = 110`
	`110 - 95 = 15`
	`15 + 24 = 39 (This works!)`
	
	`Let me verify:`
	`51 + 59 = 110`
	`110 - 95 = 15`
	`15 + 24 = 39`
	`</think>`
	`<answer> (51 + 59 - 95) + 24 </answer>`

Result	**Success** (Correctly consumed all digits exactly once, adhered to formatting tokens, and successfully bypassed the sparse reward barrier by leveraging the teacher's structural grouping suggestion).

Why the Method Works (Scaffolding Impact): In complex 4-number tasks, flat RLOO stalls early because finding a correct path randomly is highly improbable. As shown in Table 3, the teacher's hint successfully instills the concept of adding two numbers and then subtracting. Instead of searching chaotically, the RLOO loop is primed to explore adding two numbers and subtracting within its exploration inside the `<think>` block. When the first attempt ($24 + 51 - 59$) fails, this instilled concept keeps the policy on a productive path. The student easily pivots to a secondary pairing ($51 + 59 - 95 = 15$) and adds the remaining 24 to hit the target. The hint directly guides the student to learn and explore independent reasoning loops.

5.2.2 Failure Modes and Ablation Insights

Despite outperforming the baselines, qualitative analysis highlighted that:

1. **Teacher Zero-Shot Blindness:** The teacher currently operates zero-shot and strictly appends hints at the end of the problem. It does not observe or dynamically intervene during the student's actual step-by-step rollout.

6 Discussion

A key limitation is that the teacher currently operates in zero-shot mode, appending hints strictly at the end of the problem. It does not observe or dynamically intervene during the student's actual step-by-step rollout. The framework is also highly sensitive to the schedule of the hint-probability hyperparameter. Furthermore, we encountered severe training difficulties early on: using RLOO for both the teacher and student models simultaneously is unstable and leads to optimization collapse. We resolved this by using the teacher as an SFT engine to ground the loop.

Our findings show that our framework indeed was able to outperform the baseline in terms of training speed and accuracy. The teacher model was able to instill enriched hints to the student. The student learns from these hints but doesn't rely on them during test time. Future work will allow the teacher to see the student's rollout from a given hint

7 Conclusion

In conclusion, the result show our teacher-student framework with dynamic outperforms the baseline in accuracy. The teacher instills enriched hints, successfully breaking the cold start problem across progressive tiers. Crucially, the student learns from these hints but doesn't rely on them at test time, demonstrating its reasoning skills. The significance of this work is that dynamic language scaffolding is a highly effective way to smooth out optimization in sparse-reward environments where traditional RL loops easily stall. For future directions, we plan to allow the teacher model to dynamically observe the student's step-by-step rollout to provide mid-sequence hints and integrate a verifier to eliminate false-positive rewards caused by accidental semantic convergence.

8 Team Contributions

- **Group Member 1:** Luan Lam: Entire final project

Changes from Proposal Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh.

References

Avi Singh, Aviral Kumar, Sergey Levine, and Chelsea Finn. 2025. Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning. In *Proceedings of the 39th International Conference on Neural Information Processing Systems* (Vancouver, Canada) *(NeurIPS '25)*. Curran Associates, Inc., 14 pages.

Xinyu Wang, Zhihan Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Self-Evolving Curriculum for LLM Reasoning. *ACM Transactions on Asian and Low-Resource Language Information Processing* 24, 3 (2025), 411–432. doi:10.1145/3719456

Rui Zheng, Shizhe Chang, Chang Liu, and Xuanjing Huang. 2026. Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability. In *Proceedings of the 14th International Conference on Learning Representations* (Vienna, Austria) *(ICLR '26)*. 18 pages.