Extended Abstract

Motivation Fine-tuning of language models for mathematical reasoning is highly sensitive to the prompt distribution used for rollouts. In Countdown, easy prompts quickly saturate while harder prompts often produce uniformly failed samples, so uniform sampling can waste rollout budget on examples that provide little learning value. We study whether the training distribution itself can be adapted during RLOO fine-tuning without changing the underlying policy-gradient objective.

Method We introduce Advantage-Driven Synthetic Curriculum (ADSC), a curriculum learning layer for REINFORCE Leave-One-Out (RLOO). ADSC splits Countdown prompts and synthetically-generated prompts into difficulty tiers and uses a multi-armed bandit router to sample from these difficulty buckets. The router reward is the mean absolute RLOO advantage where buckets receive high value when the student has mixed rollout outcomes, and low value when all sampled completions either succeed or fail. Therefore, ADSC uses a signal already computed by RLOO to identify prompts near the student's current learning frontier.

Implementation ADSC keeps the default RLOO rollout, reward, advantage, policy update, and checkpoint logic intact, while changing the prompt selection and generation, as well as the batch construction. Real prompts come from the original Countdown training set, while synthetic prompts are generated online either by a verified rule-based expression-tree generator or by a Qwen3-8B teacher model. All synthetic tasks are filtered by an exact Countdown verifier, checked for solvability, matched to the requested difficulty tier, and deduplicated before entering training.

Experiments We compare four RLOO-based variants under the same 75-step training budget: vanilla RLOO with uniform real-data sampling, RLOO with a real-only MAB curriculum, ADSC with rule-based synthetic tasks, and ADSC with Qwen3-8B teacher-generated synthetic tasks. We evaluate on 50 held-out 3/4-number Countdown prompts with 16 sampled responses per prompt, reporting mean verifier score, exact completion rate, and Pass@16. We also test out-of-distribution generalization on 50 held-out 5-number prompts.

Results In this short-budget comparison, real-only MAB improves over vanilla RLOO, increasing mean verifier score from 0.5156 to 0.5247 and Pass@16 from 68% to 70%. Qwen-teacher ADSC achieves the highest measured result, with 0.5280 mean score, 50.75% exact completion, and 72% Pass@16. Rule-based ADSC reaches 0.5025 mean score, 47.75% exact completion, and 68% Pass@16, suggesting that verified synthetic data is not automatically beneficial unless the generated task distribution is useful for the current student. The largest remaining in-distribution gap is task difficulty: Qwen-teacher ADSC reaches 91.7% Pass@16 on 3-number tasks but only 53.8% Pass@16 on 4-number tasks.

Discussion The results suggest that RLOO advantage is a useful curriculum signal and that adaptive routing alone is a strong short-budget baseline. Teacher-generated synthetic tasks provide a further improvement in Pass@16 in this setting, while purely procedural generation is less effective. However, OOD generalization remains limited: on held-out 5-number prompts, neither synthetic ADSC student solves any prompt exactly at Pass@16. This indicates that broad tier-level curricula improve in-distribution learning but are not sufficient for harder compositional transfer.

Conclusion ADSC shows that rollout advantages produced by RLOO can be reused as a low-cost curriculum signal. Rather than introducing a new RL objective, ADSC improves the training process by adaptively choosing which real and synthetic prompts the student practices. Future work should scale training, run multiple seeds, target recent student failures with the teacher model, and incorporate denser process-level feedback for longer arithmetic chains.

Advantage-Driven Synthetic Curriculum for Reinforcement Learning based Fine-Tuning of Large Language Models

Nate Demchak
Department of Computer Science
Stanford University
natedemchak@stanford.edu

Pravin Ravishanker
Department of Computer Science
Stanford University
pravinr@stanford.edu

Oscar Li
Department of Computer Science
Stanford University
oscarli@stanford.edu

Abstract

We introduce Advantage-Driven Synthetic Curriculum (ADSC), a drop-in extension to REINFORCE Leave-One-Out (RLOO) policy-gradient training for reinforcement learning based fine-tuning of large language models. ADSC keeps the RLOO objective fixed and instead changes the prompt distribution used for rollouts. It uses a multi-armed bandit router over difficulty buckets, where each bucket is scored by the student's mean absolute RLOO advantage, a signal already computed during training. We compare vanilla RLOO, RLOO with a real-only MAB curriculum, ADSC with rule-based synthetic task generation, and ADSC with Qwen3-8B teacher-generated synthetic tasks on the Countdown arithmetic task. Under a 75-step training budget, Qwen-teacher ADSC achieves the highest measured Pass@16 and mean verifier score in our comparison, with 0.5280 mean score, 50.75% exact completion, and 72% Pass@16. Vanilla RLOO achieves 0.5156 mean score, 48.00% exact completion, and 68% Pass@16, while real-only MAB routing improves to 0.5247 mean score, 50.75% exact completion, and 70% Pass@16. These results suggest that advantage-driven routing is a strong short-budget baseline and that teacher-generated synthetic tasks can provide an additional improvement in Pass@16. However, OOD 5-number evaluation shows that stronger compositional transfer remains unsolved in this setting.

1 Introduction

In this project, we use reinforcement learning in order to improve the performance of Large Language Models that solve the Countdown problem. Given a set of numbers and a target value, the LLM aims to find and report the sequence of arithmetic operations that transform the given numbers into the target. Curriculum learning is an approach that has long been used to improve training efficiency by showing models progressively harder tasks (Bengio et al. , 2009). It has grown increasingly relevant in the context of reinforcement learning (RL) for post-training large language models (LLMs), where rule-based rewards have incentivized complex reasoning.

To address the challenges of sparse rewards and low-success regions in mathematical reasoning tasks like Countdown, we propose the Advantage-Driven Synthetic Curriculum (ADSC). This framework shifts the training paradigm from a static dataset to an automated curriculum tailored

for computationally constrained models. Preliminary training logs suggested a recurring reasoning failure: as task complexity increases, the model often "drifts," losing track of basic order of operations and fundamental mathematical constraints (such as the properties of integers and whole numbers) during extended thinking chains. To counteract this, we decompose the Countdown task into discrete difficulty tiers based on integer count and target complexity, gradually transitioning the model's focus to solidify foundational logic before exposing it to advanced, sparse-reward environments.

Our primary technical contribution is a compute-efficient, self-updating Multi-Armed Bandit router. While existing methods often treat curriculum generation and task routing as separate or computationally heavy processes, our framework unifies them within the existing RLOO pipeline. We contribute a routing mechanism that uses the model's internal RLOO policy-gradient signals (absolute advantage) as the MAB reward for both real and synthetic tasks. By doing so, we combine adaptive task selection with verified synthetic task generation, while avoiding the overhead of training a separate meta-RL teacher policy. Under a short training budget, this allows a 0.5B parameter student model to spend more rollout budget on examples near its current learning frontier using standard on-policy gradient updates.

To evaluate ADSC, we compare four RLOO-based variants: vanilla RLOO with uniform real-data sampling, RLOO with a real-only MAB curriculum, ADSC with verified rule-based synthetic task generation, and ADSC with Qwen3-8B teacher-generated synthetic tasks. This comparison separates three effects: the value of adaptive routing alone, the value of adding synthetic tasks, and the value of using a stronger teacher model to generate those synthetic tasks. Under the same 75-step training budget, real-only MAB improves over vanilla RLOO from 0.5156 to 0.5247 mean verifier score and from 68% to 70% Pass@16, while Qwen-teacher ADSC reaches the highest measured Pass@16 of 72%. These gains are modest but suggestive under the short compute budget. We also evaluate out-of-distribution 5-number prompts to test whether in-distribution curriculum gains transfer to longer compositional arithmetic problems.

2 Related Work

In this report, we examine three recent papers about curriculum learning. These methods include using a predefined difficulty scheduler (Parashar et al., 2025), adaptive scheduling via multi-armed bandits (Chen et al., 2025), and meta-RL with grounded teacher rewards (Sundaram et al., 2026). Parashar et al. (2025) propose E2H Reasoner, which schedules training examples over labeled difficulty tiers from easy to hard. The schedule only depends on the training step rather than the model's current success rate. Chen et al. (2025) make the curriculum adaptive with SEC, which chooses which difficulty category to sample from as a multi-armed bandit problem, favoring categories where the model is currently learning the most. SEC adapts as the model improves, but since it draws from a predefined set of categories, it has nothing to act on when the success rate of every category is 0. Sundaram et al. (2026) address this edge case with SOAR, in which a teacher copy of the model generates stepping stones and is rewarded based on how much the student improves on the hard held-out problems. SOAR continues to make progress where direct RL would otherwise plateau, but at a substantial compute cost.

Altogether, these methods cover complementary aspects of curriculum learning, but each one introduces a specific limitation: predefined schedules ignore the model's current success rate, bandit-based methods stall when no category is learnable, and meta-RL teachers are expensive to train. Our approach, which uses an Advantage-Driven Synthetic Curriculum for RLOO based fine-tuning of Large Language Models, is designed to reduce these limitations by reusing an existing RLOO signal for adaptive routing and by adding verified synthetic tasks without training a separate teacher policy.

3 Method

Our method, Advantage-Driven Synthetic Curriculum (ADSC), augments the standard RLOO training pipeline with an adaptive prompt-selection layer. The policy objective is left entirely unchanged; ADSC only modifies which prompts are used to construct each rollout batch. We describe each component in turn, justify the design choices, and situate them relative to prior work.

3.1 Background: RLOO

We build on REINFORCE Leave-One-Out (RLOO), an online policy-gradient algorithm used in the default Countdown fine-tuning pipeline with a rule-based verifier reward. RLOO reduces the variance of the standard REINFORCE estimator by constructing a per-sample baseline from the other rollouts sampled from the same prompt. Concretely, given k i.i.d. rollouts $y_{(1)}, \ldots, y_{(k)} \sim \pi_\theta(\cdot \mid x)$, the gradient estimate is:

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y_{(i)}, x) - \frac{1}{k-1}\sum_{j\neq i} R(y_{(j)}, x)\right]\nabla \log \pi(y_{(i)} \mid x) \tag{1}$$

The bracketed term is the RLOO advantage for sample i: the reward of that rollout minus the mean reward of all other rollouts from the same prompt. This baseline is unbiased—it does not depend on $y_{(i)}$—and requires no separate value network, making RLOO computationally attractive for LLM fine-tuning.

Importance weighting. In practice, rollouts are generated using vLLM for throughput, while the policy-gradient update is computed using the HuggingFace model being fine-tuned. Because these two execution paths can yield slightly different token-level log probabilities due to kernel-specific numerics or tokenization edge cases, we treat vLLM as a behavior policy μ and the HuggingFace model as the target policy π_θ. We apply a sequence-level importance ratio to reduce mismatch between the rollout engine and update model:

$$w(y, x) = \exp(\log \pi_\theta(y \mid x) - \log \mu(y \mid x)) \tag{2}$$

Each RLOO advantage term is multiplied by $w(y, x)$. For numerical stability the ratio is computed in log-space and clipped to a maximum value. Because the ratio is clipped, this correction should be viewed as a practical stability tradeoff rather than a strictly unbiased off-policy estimator.

Role in ADSC. ADSC leaves this entire objective unchanged. The advantage estimates $\hat{A}(y_{(i)} \mid x)$ produced by RLOO are reused by the MAB router as a learning-progress signal (Section 3.3), requiring no additional forward passes or auxiliary models. The following equations show how the advantage is calculated for a single training example in a difficulty bucket consisting of K samples. A useful bucket for ADSC is one in which a student has mixed outcomes, rather than a bucket where all presented problems are either easy or hard. Low absolute advantages mean that all rollouts have similar rewards. High absolute advantages mean reward varies across rollouts, so this bucket should be prioritized during sampling.

$$\hat{A}(y_k \mid x) = r(y_k, x) - \frac{1}{K-1}\sum_{j\neq k} r(y_j, x) \qquad s(x) = \frac{1}{K}\sum_{k=1}^{K} |\hat{A}(y_k \mid x)| \tag{3}$$

For binary verifier rewards, this signal has a simple interpretation. If the student succeeds on prompts from a bucket with probability p, then the expected absolute advantage is largest when successes and failures are mixed and approaches zero when the bucket is either uniformly solved or uniformly failed. Thus, mean absolute RLOO advantage acts as an inexpensive proxy for frontier difficulty.

3.2 Difficulty Bucketing

We partition the training prompt pool into four difficulty tiers based on the structural features of each Countdown problem. An out-of-distribution (OOD) tier is reserved for evaluation only.

3.3 Multi-Armed Bandit Router

ADSC defines **eight MAB arms**, one for each element of $\{\text{real data, synthetic data}\} \times \{\text{T0}, \text{T1}, \text{T2}, \text{T3}\}$. At each training step the router samples a probability vector over arms, selects an arm combination, and then draws prompts that correspond to the chosen difficulty bucket and real/synthetic option to fill the data batch.

Table 1: Curriculum difficulty buckets.

Bucket	Description	Source
T0	3 numbers, small targets, easiest tasks	train
T1	3–4 numbers, target ≤ 75	train
T2	4 numbers, target ≤ 150	train
T3	4 numbers, larger or sparse targets	train
OOD	5-number tasks and sparse variants	eval only

Arm reward signal. The bandit reward for arm c is the **mean absolute advantage** of rollouts drawn from that arm:

$$s(x) = \frac{1}{K} \sum_{k=1}^{K} \left| \hat{A}(y_k \mid x) \right|, \qquad |A|_c = \frac{1}{|B_c|} \sum_{x \in B_c} s(x) \tag{4}$$

where B_c is the set of prompts assigned to arm c in the current batch. $|A|_c$ is high when rollout rewards are mixed, meaning some succeed and some fail, placing the arm in the model's current learning zone. It is low in two degenerate cases: (i) all rollouts succeed (bucket saturated) or (ii) all rollouts fail (bucket too hard). Using absolute advantage as a progress signal rather than raw reward prevents the router from chasing easy arms.

Q-value update. Each arm maintains a scalar Q-value updated via exponential moving average (EMA) after each training step:

$$Q_{t+1}(c) = (1 - \alpha)\, Q_t(c) + \alpha\, |A|_c \tag{5}$$

where α is the EMA learning rate. We use $\alpha = 0.15$ in all ADSC curriculum experiments.

Arm selection. Arm probabilities follow an ε-softmax policy:

$$p(c) = (1 - \varepsilon) \frac{\exp(Q(c)/\tau)}{\sum_{c' \in \mathcal{A}} \exp(Q(c')/\tau)} + \frac{\varepsilon}{|\mathcal{A}|} \tag{6}$$

where τ is a temperature controlling the sharpness of exploitation and ε is an exploration floor that ensures every arm retains a nonzero selection probability.

Routing constraints. Three hard constraints prevent degenerate routing:

1. Real training data $\geq 30\%$ of each batch (prevents policy drift on synthetic data alone).
2. Synthetic data $\leq 50\%$ of each batch (preserves grounding in the original distribution).
3. Hard-tier arms (T2, T3) remain active after a warmup period (prevents the router from permanently settling on easy arms during early training).

Novelty relative to prior work. Prior curriculum methods for LLMs use either fixed easy-to-hard schedules or heuristic competence measures such as loss thresholds Bengio et al. (2009). Graves et al. Graves et al. (2017) also use a MAB over task buckets, but with a supervised loss-reduction signal. Our key novelty is using the RLOO absolute advantage—a quantity already computed during training at no extra cost—as the routing reward. This provides a self-updating measure of learning progress directly tied to the policy gradient objective, and it operates within the RL fine-tuning regime without requiring an additional supervised signal.

3.4 Synthetic Task Generation

When the router selects a synthetic arm, ADSC generates a new Countdown task on-the-fly for the selected tier and passes it through a verifier before including it in the batch.

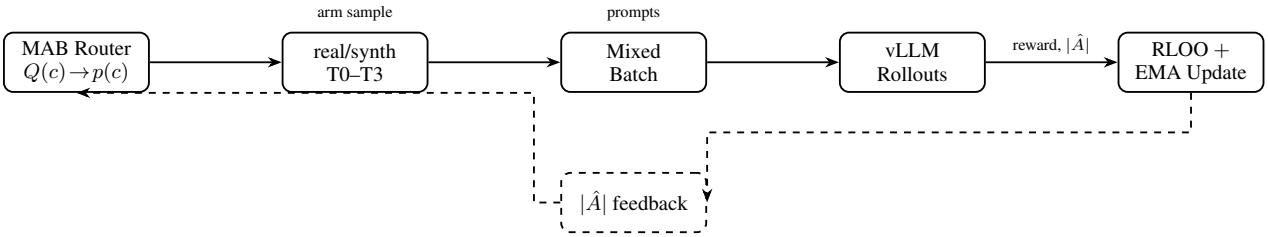

Figure 1: ADSC training loop. The MAB router samples arm probabilities $p(c)$ from Q-values, assembles a mixed batch of real and synthetic prompts, runs RLOO rollouts via vLLM, updates the policy, and feeds mean absolute advantage $|\hat{A}|$ back to the router.

Rule-based generator. Numbers are sampled uniformly for the target tier, assembled into a random binary expression tree with operators drawn from $\{+, -, \times, \div\}$, and evaluated using exact `Fraction` arithmetic to produce an integer target. Solvability is guaranteed by construction since the target is the output of the expression.

Qwen3-8B teacher. The teacher LLM is prompted with the tier specification and asked to propose a valid Countdown task. Because the teacher can produce invalid outputs, every candidate passes through the same verifier described below.

Verification pipeline. A generated task is accepted if and only if:

- The target is an integer.
- The task is solvable (confirmed by exhaustive search over operator assignments).
- The number count and target magnitude are consistent with the declared tier.
- The pair (`sorted numbers`, `target`) has not been seen before (deduplication).

Justification. Online generation addresses a data scarcity problem: real training data for hard tiers is finite and the model can memorize it. An unlimited stream of novel verified tasks in precisely the tiers the router targets prevents overfitting and allows the curriculum to remain challenging throughout training. Teacher-generated tasks additionally expose the student to more naturalistic difficulty distributions, since the teacher implicitly biases toward number–target combinations that are arithmetically non-trivial.

3.5 ADSC Training Loop

The complete procedure is summarized in Algorithm 1 and illustrated in Figure 1.

Algorithm 1 ADSC Training Loop

Require: Base model π_θ, real prompt pool $\mathcal{D} = \{\mathcal{D}_{T0}, \ldots, \mathcal{D}_{T3}\}$, arms $\mathcal{A} = \{\text{real}, \text{synth}\} \times \{T0, \ldots, T3\}$, batch size B, rollouts per prompt K, EMA rate α, temperature τ, exploration ε

Initialize $Q(c) \leftarrow 0$ for all $c \in \mathcal{A}$
for $t = 1, 2, \ldots, T$ **do**
  // Routing
  Compute $p(c)$ via Eq. (6); apply hard constraints
  Sample arm assignments for B prompt slots according to $p(c)$
  // Batch construction
  for each slot assigned to a *real* arm c **do**
    Sample $x \sim \mathcal{D}_{\text{tier}(c)}$
  end for
  for each slot assigned to a *synthetic* arm c **do**
    repeat
      Generate candidate x' (rule-based or teacher) for tier(c)
      Run verifier on x'
    until x' passes verification
    $x \leftarrow x'$
  end for
  // Rollout & policy update
  for each prompt x in batch **do**
    Sample $\{y_1, \ldots, y_K\} \sim \pi_\theta(\cdot \mid x)$ via vLLM
    Compute rewards $\{r(y_k, x)\}$ and RLOO advantages $\{\hat{A}(y_k \mid x)\}$ via Eq. (1)
  end for
  Apply importance weights (Eq. (2)); update θ via RLOO gradient
  // Router update
  for each arm c that received prompts **do**
    $|A|_c \leftarrow \frac{1}{|B_c|} \sum_{x \in B_c} s(x)$ ▷ Eq. (4)
    $Q(c) \leftarrow (1 - \alpha)\, Q(c) + \alpha\, |A|_c$ ▷ Eq. (5)
  end for
end for

4 Experimental Setup

Each prompt specifies a target integer and a set of numbers. The model must produce an arithmetic expression that uses each number exactly once and evaluates to the target. Responses are scored by a rule-based verifier where exact valid equations receive reward 1.0, parseable but incorrect answers receive partial reward, and invalid responses receive reward 0.0.

Base model and training objective. All experiments start from the same SFT checkpoint, `asingh15/qwen-sft-countdown-defaultproj`, with Qwen2.5-0.5B as the tokenizer. The student is trained with the existing RLOO pipeline. ADSC does not modify the RLOO objective, reward function, rollout generation, tokenization, optimizer step, or checkpointing code; it only changes which prompts are sampled into each training batch.

Training data. The real training prompts are drawn from `asingh15/countdown_tasks_3to4`. ADSC partitions these prompts into difficulty tiers using the ground-truth numbers and target. Synthetic prompts are generated online for the selected tier. The rule-based generator samples numbers, constructs a random arithmetic expression tree, evaluates it with exact arithmetic (using Python's built-in Fraction library), and uses the resulting integer as the target. The Qwen-teacher generator prompts Qwen3-8B to propose tasks for the requested tier. In both cases, generated tasks are accepted only if they pass the exact verifier, match the requested tier, and are not duplicates.

Curriculum variants. We compare four RLOO-based training variants:

1. **Vanilla RLOO**: uniform sampling from the original real Countdown training set, with no curriculum and no synthetic data.

2. **RLOO + real-only MAB**: the ADSC bandit router samples among real difficulty buckets only, isolating the effect of adaptive prompt routing without synthetic data.
3. **ADSC rule-based**: the full ADSC router samples from both real buckets and verified rule-based synthetic buckets.
4. **ADSC Qwen-teacher**: the full ADSC router samples from both real buckets and Qwen3-8B-generated synthetic buckets.

Router configuration. For ADSC variants, the router arms are difficulty/source buckets. In the full synthetic setting, these are $\{\text{real}, \text{synth}\} \times \{T0, T1, T2, T3\}$. The router maintains an EMA value $Q(c)$ for each arm and updates it using mean absolute RLOO advantage from prompts sampled from that arm. We use router EMA rate $\alpha = 0.15$, softmax temperature $\tau = 0.25$, a minimum real-data fraction of 0.30, a maximum synthetic-data fraction of 0.50, and a hard-tier quota of 0.10 after warmup.

Training budget. Each model is trained for 75 global steps with group size 8, batch size 128, gradient accumulation over 128 examples, learning rate 1×10^{-5}, temperature 1.0, top-p 1.0, and top-$k = -1$. Checkpoints are saved every 25 steps, and all reported results use the global-step-75 checkpoint.

Evaluation. For in-distribution evaluation, we use 50 held-out 3/4-number Countdown prompts and sample 16 responses per prompt. We report mean verifier score, exact completion rate across all sampled responses, and Pass@16, where a prompt is counted as solved if any of the 16 responses is exactly correct. We also break results down by 3-number and 4-number prompts. For out-of-distribution evaluation, we use 50 held-out 5-number prompts and report Exact Pass@16 and nonzero/partial Pass@16.

5 Results

We evaluate Advantage-Driven Synthetic Curriculum (ADSC) on the Countdown arithmetic task under the same 75-step training budget for all methods. We compare four RLOO-based methods: vanilla RLOO, RLOO with a real-only MAB curriculum, ADSC with rule-based synthetic task generation, and ADSC with Qwen3-8B teacher-generated synthetic task generation. The held-out in-distribution evaluation contains 50 3/4-number Countdown prompts, with 16 sampled responses per prompt. We report mean verifier score, exact completion rate across all sampled responses, and Pass@16.

5.1 Baseline Comparison

Table 2 compares vanilla RLOO, real-only adaptive routing, and the two ADSC synthetic curriculum variants. This comparison is designed to separate the effect of adaptive routing alone from the effect of combining routing with synthetic task generation.

Table 2: Default Countdown Evaluation on 50 Held-Out 3/4-Number Prompts

Student	Mean Score	Exact Rate	Pass@16
Vanilla RLOO	0.5156	48.00%	68%
RLOO + real-only MAB	0.5247	**50.75%**	70%
ADSC rule-based	0.5025	47.75%	68%
ADSC Qwen-teacher	**0.5280**	**50.75%**	**72%**

The vanilla RLOO baseline measures the effect of the original uniform prompt distribution. Real-only MAB improves mean score from 0.5156 to 0.5247 and Pass@16 from 68% to 70%, suggesting that advantage-driven routing over real difficulty buckets is useful even without synthetic data. Qwen-teacher ADSC achieves the highest mean score and Pass@16 in this comparison, suggesting that teacher-generated synthetic tasks can add value beyond real-only adaptive routing. Rule-based ADSC does not outperform vanilla RLOO in this short-budget setting, indicating that synthetic data quality is important: simply adding verified procedural tasks is not sufficient.

5.2 Performance by Difficulty

Table 3 breaks performance down by 3-number and 4-number tasks. Across all variants, 3-number prompts are substantially easier than 4-number prompts, confirming that longer arithmetic chains remain the main bottleneck.

Table 3: Difficulty Breakdown: 3-Number vs. 4-Number Tasks

Student	3-num Exact	3-num Pass@16	4-num Exact	4-num Pass@16
Vanilla RLOO	70.6%	87.5%	27.2%	50.0%
RLOO + real-only MAB	70.3%	87.5%	**32.7%**	53.8%
ADSC rule-based	70.3%	87.5%	26.9%	50.0%
ADSC Qwen-teacher	**72.4%**	**91.7%**	30.8%	**53.8%**

The 3-number results are tightly clustered for vanilla RLOO, real-only MAB, and rule-based ADSC, while Qwen-teacher ADSC improves both exact rate and Pass@16. The larger differences appear on 4-number tasks. Real-only MAB achieves the best 4-number exact rate at 32.7%, and Qwen-teacher ties for the best 4-number Pass@16 at 53.8%. This suggests that adaptive routing is especially helpful on harder in-distribution tasks, while teacher-generated synthetic tasks improve Pass@16 on easier and mixed-difficulty prompts.

5.3 Out-of-Distribution Generalization

Finally, we evaluated the models on an Out-of-Distribution (OOD) dataset comprising 50 strictly held-out 5-number prompts to test for compositional transfer.

Table 4: Out-of-Distribution Evaluation on Held-Out 5-Number Prompts

OOD Student	Exact Pass@16	Nonzero Pass@16
Rule-based ADSC	0%	90%
Qwen-teacher ADSC	0%	82%

The results in Table 4 expose an important failure mode: neither the rule-based ADSC student nor the Qwen-teacher ADSC student was able to solve a single 5-number prompt exactly, resulting in a 0% Exact Pass@16 rate. Interestingly, both models maintained a high Nonzero Pass@16 rate (90% for rule-based and 82% for Qwen-teacher). This discrepancy suggests that the models often produce partially valid or parseable responses, but still fail to reliably execute the arithmetic logic required for extended, unseen problem lengths.

6 Discussion, Limitations, and Future Work

The baseline experiments clarify what ADSC contributes beyond the default RLOO pipeline. Vanilla RLOO tests whether the original uniform training distribution is sufficient under the same short compute budget. The real-only MAB baseline suggests that advantage-driven routing alone is useful: it improves mean score and Pass@16 over vanilla RLOO without introducing any synthetic prompts. The comparison between real-only MAB and the two ADSC variants then isolates the role of synthetic data quality. Rule-based synthetic generation underperforms the stronger real-only MAB baseline, while Qwen-teacher ADSC achieves the highest mean score and Pass@16 in this comparison. This suggests that adaptive routing is an important mechanism, but synthetic tasks only help when the generator provides useful training examples.

The empirical results suggest that ADSC can improve in-distribution performance on 3- and 4-number tasks by intelligently routing prompt samples during RLOO fine-tuning. However, our experiments also surfaced several limitations related to model capacity, curriculum generation, and compositional transfer that motivate future work.

Teacher Model Capacity Limitations: During the initial phases of the project, our attempts to utilize a smaller, 0.5B parameter model for dynamic task generation ultimately failed. We observed

that the 0.5B model lacked the inherent zero-shot arithmetic reasoning capabilities necessary to serve as an effective curriculum generator. This deficiency led to unacceptably high hallucination rates and the frequent generation of invalid, unsolvable target numbers, particularly when attempting to synthesize problems for the harder T2 and T3 difficulty tiers. This directly motivated our shift to the more capable Qwen3-8B model, suggesting that teacher capacity matters for autonomous task generation in logic-based mathematical domains.

Scaling and Rigorous Evaluation: The current evaluation results are constrained by a relatively short training horizon of 75 global steps. While sufficient to compare the Multi-Armed Bandit router and the Qwen-teacher under a fixed budget, this budget does not reveal the higher-budget behavior of the models. To better validate compositional transfer and stabilize the observed learning dynamics, future work should support full-length training runs across multiple random seeds. This scaled approach, alongside the addition of permanent 5-number OOD evaluations, could be paired with the upgraded 8B teacher model to facilitate a more robust long-term asymmetric self-play environment.

Targeted Exploitation of Weaknesses: Finally, we intend to refine the curriculum's long-term routing strategy. As the student's policy matures and saturates the easier T0 and T1 buckets, the curriculum may need to shift from broad, exploratory task sampling to the targeted exploitation of specific student weaknesses. We will explicitly prioritize routing toward low-density, complex tasks that force strict deductive logic over random operator guessing. This targeted exploitation will be crucial for testing whether the student can develop generalized, compositional mathematical skills capable of out-of-distribution transfer, rather than relying on memorized structural shortcuts.

7 Conclusion

In this work, we addressed the inefficiency of uniform prompt sampling in reinforcement learning fine-tuning for mathematical reasoning tasks. We introduced Advantage-Driven Synthetic Curriculum (ADSC), a drop-in extension to the standard RLOO pipeline that uses a multi-armed bandit router to dynamically select prompts across real and synthetic difficulty buckets. ADSC keeps the RLOO objective fixed and instead reuses the student's mean absolute RLOO advantage as a self-updating curriculum signal, allowing the trainer to spend more rollout budget on buckets where the current policy has mixed success and failure.

Our baseline comparison suggests that adaptive routing itself is useful. Vanilla RLOO achieves 0.5156 mean verifier score, 48.00% exact completion, and 68% Pass@16 on held-out 3/4-number Countdown prompts. RLOO with a real-only MAB curriculum improves to 0.5247 mean score, 50.75% exact completion, and 70% Pass@16, suggesting that advantage-driven routing over real difficulty buckets can improve short-budget training without adding synthetic data. The highest measured result is Qwen-teacher ADSC, which achieves 0.5280 mean score, 50.75% exact completion, and 72% Pass@16. In contrast, rule-based ADSC does not outperform vanilla RLOO in this setting, suggesting that synthetic tasks only help when their distribution is useful for the student's current learning frontier.

Despite these in-distribution improvements, our results highlight the persistent difficulty of compositional arithmetic generalization. Performance drops sharply from 3-number to 4-number tasks, and both synthetic ADSC variants achieve 0% exact Pass@16 on out-of-distribution 5-number prompts. This indicates that broad tier-level routing and outcome-only verifier rewards are not sufficient for robust transfer to longer reasoning chains.

Future work should scale the training horizon, evaluate multiple seeds, and move from broad bucket-level sampling toward teacher generation that targets recent student failure modes. Incorporating denser process-level feedback may also be necessary for teaching longer multi-step arithmetic reasoning. Overall, ADSC suggests that managing the rollout prompt distribution is an important lever for RL fine-tuning, complementary to the choice of policy-gradient objective.

8 Team Contributions

- **Group Member 1: Nate Demchak** Implemented the ADSC extension, including the curriculum router, rule-based synthetic task generator, and Qwen3-8B teacher generation path.

Helped code up the RLOO (Reinforce Leave One Out) portion of the project. Contributed to Extended Abstract and Experimental Setup sections of report.

- **Group Member 2: Oscar Li** Implemented the Preference Optimization (IPO) code portion. Conducted preliminary experiments evaluating the 0.5B teacher model within the RLOO framework, and contributed to results, limitations, and conclusion section of the final report.
- **Group Member 3: Pravin Ravishanker** Coded up the Supervised Fine Tuning (SFT) portion of the project. Coded up the RLOO (Reinforce Leave One Out) portion of the project. Contributed to Introduction, Related Work, and Methods sections of report.

Changes from Proposal Compared with the original proposal, we implemented ADSC as a lightweight extension on top of the completed RLOO pipeline rather than a full SOAR-style meta-RL teacher. We kept the RLOO objective fixed and focused on adaptive prompt routing, verified synthetic task generation, and teacher-generated Countdown tasks. We also added 5-number OOD evaluation to test whether curriculum gains transfer to harder compositional settings.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning* (2009).

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. 2025. Self-Evolving Curriculum for LLM Reasoning. arXiv:2505.14970 [cs.LG]

Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated Curriculum Learning for Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1311–1320.

Aakriti Parashar, Lin Gui, Yixin Li, et al. 2025. E2H Reasoner: Curriculum Reinforcement Learning from Easy to Hard Tasks Improves LLM Reasoning. arXiv:2506.06632 [cs.LG]

Shobhita Sundaram, John Quan, Ariel Kwiatkowski, Kartik Ahuja, Yann Ollivier, and Julia Kempe. 2026. Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability. arXiv:2601.18778 [cs.LG]