# Extended Abstract

**Motivation** Online policy-gradient methods such as RLOO Ahmadian et al. (2024) fine-tune reasoning models on a *fixed* distribution of problems. As the policy improves, most problems drift out of usefulness — easy ones are always solved (zero gradient) and hard ones are always failed (no reward signal). The problems that drive learning are those the model solves about half the time. We ask whether a model can learn to generate its own curriculum, automatically proposing problems at the edge of its current ability.

**Method.** We co-train two agents on the Countdown arithmetic task Gandhi et al. (2024). A *Solver* (Qwen2.5-0.5B, fine-tuned with RLOO) attempts the problems and a tiny, non-LM *Proposer* ($\sim$70k params) outputs a distribution over problem difficulty knobs, from which a rule-based generator samples solvable instances. The Proposer is rewarded by information gain $p(1-p)$, the variance of the Solver's binary success outcome, which is maximized when the Solver's pass rate $p$ is $0.5$ and which dictates the magnitude of the RLOO gradient. A policy-entropy diversity bonus discourages the curriculum from collapsing onto a single difficulty.

**Implementation.** We build on the current RLOO stack (importance-weighted RLOO update, rule-based Countdown verifier), replacing only the static prompt source with the Proposer. The Proposer is trained with the same RLOO estimator on its batch of proposed problems. We also implement two baselines as ablations (uniform-random and hand-scheduled problem generation) and a mechanism to mix curated original dataset problems into the Solver's batch.

**Results.** The learned curriculum *underperforms* the RLOO milestone in-distribution (i.e. 3 to 4 input numbers) (40.5% vs 59.0% average accuracy) and the Solver's accuracy *decreases* monotonically with increasing curriculum training step. The cause is *Proposer collapse*: with a weak diversity bonus ($\lambda$=0.01), the Proposer's entropy falls to zero and it concentrates on a single hard difficulty, so the Solver overfits a narrow band of problems. Two targeted fixes recover the method: strengthening the diversity coefficient ($\lambda$=0.1–0.3) lifts accuracy to $\sim$51% (matching the uniform-generation baseline), and additionally mixing 50% curated data into each batch reaches 58.3% average accuracy—matching the RLOO milestone (59.0%) and modestly exceeding it on pass@$k$. Out-of-distribution generalization to harder problems (5 to 6 input numbers) remains at the floor for all methods, indicating the capability gap exceeds what any training-distribution intervention is capable of fixing at this scale.

**Discussion.** The information-gain objective is insufficient on its own. The diversity term is not just a regularizer but a critical stabilizer of the entire method. The two fixes address two distinct failure modes—diversity prevents *collapse*, while curated-data mixing addresses the *distribution shift* between generated and curated problems. The situation then becomes that reaching genuinely out-of-distribution difficulty requires *generating* problems, but generation introduces a gap from the curated original data that must be actively managed.

**Conclusion.** A learned, information-gain curriculum collapses without strong diversity regularization. Once collapse and the distribution shift are addressed, a self-generated curriculum *matches* the RLOO milestone on the original curated dataset while sourcing its own problems. We provide a mechanical diagnosis and a recipe (strong diversity regularization plus curated-data mixing) for stable self-generated curricula.

# Learnable Curricula via Self-Play for Verifiable Reasoning Tasks

**Kai Wen**
Department of Computer Science
Stanford University
kwen7@stanford.edu

## Abstract

Reinforcement learning fine-tunes reasoning language models on a fixed problem distribution, but as the policy improves most problems stop carrying a learning signal. We investigate whether a model can learn its own curriculum by co-training a Solver (Qwen2.5-0.5B + RLOO) with a tiny non-LM Proposer that learns a distribution over Countdown problem difficulty, rewarded by the information-gain signal $p(1-p)$. The learned curriculum performs worst of four training-distribution conditions (fixed, uniform, hand-scheduled, learned) and degrades the Solver the more it trains. We trace this to Proposer entropy collapse. Two targeted fixes recover it: a stronger diversity (entropy) regularizer prevents collapse, and mixing curated original data into the Solver's batch closes the gap between generated and curated problems, together *matching* the RLOO milestone (58.3% vs 59.0% average accuracy) while the curriculum sources its own problems. Out-of-distribution generalization is capability-bound at this scale and fails on 5 to 6 input numbers. Our contributions are a verifier-driven learned-curriculum implementation, a mechanical analysis of why the naive information-gain objective fails, and a recipe that makes a self-generated curriculum competitive with a curated dataset.

## 1 Introduction

Reinforcement learning from verifiable rewards has become a standard tool for post-training reasoning models. Methods such as RLOO Ahmadian et al. (2024) sample candidate solutions from the current policy, score them with a verifier, and apply a policy-gradient update. The problem is that the *problem distribution is fixed*. As the policy improves, an increasing fraction of problems lack learning signal: those the model always solves yield no advantage (and hence no gradient), and those it always fails yield no reward. The problems that actually teach the model are those it solves roughly half the time—at the edge of its current ability Sundaram et al. (2026).

This motivates our question: *can a model learn to generate its own curriculum*, continually proposing problems at its current edge of learnability? We study this on Countdown Gandhi et al. (2024), an arithmetic reasoning task with a rule-based verifier, which lets us define a verifier-grounded difficulty signal. We pair a Solver (Qwen2.5-0.5B + RLOO) with a tiny, non-LM Proposer that learns a distribution over problem difficulty and is rewarded for proposing maximally informative problems.

We make three contributions: (1) We design and implement a verifier-driven self-play curriculum in which the Proposer's reward is the information gain $p(1-p)$—equivalently the variance of the Solver's binary outcome and the quantity that determines the RLOO gradient magnitude. (2) We evaluate it against fixed, uniform, and hand-scheduled problem distributions, both in-distribution and out-of-distribution. (3) We provide a mechanical analysis of a *negative* result: the learned curriculum underperforms the fixed dataset and worsens the Solver over training, which we trace to Proposer collapse. We then study diversity regularization and curated-data mixing as fixes.

Stanford CS224R 2025 Final Report

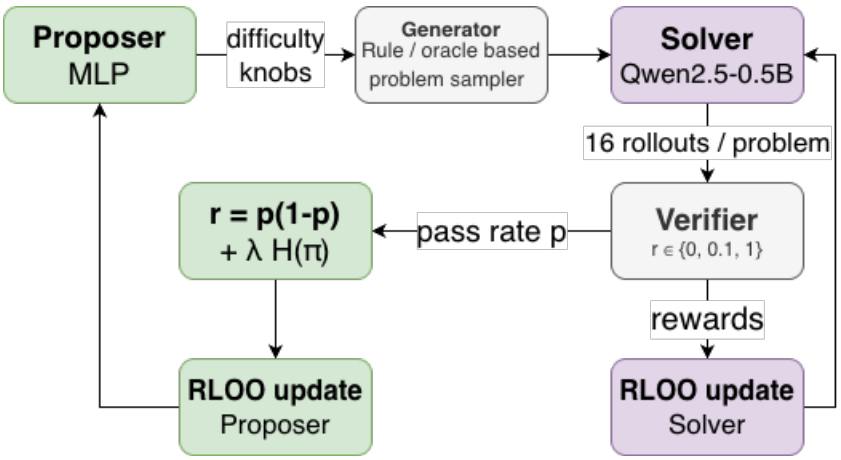


Figure 1: The co-evolution loop. The Proposer outputs difficulty knobs, a rule-based generator samples solvable problems, and the Solver produces rollouts scored by the verifier. The pass rate $p$ rewards the Proposer via $p(1-p)$ and the rollouts update the Solver via RLOO.

## 2 Related Work

**Self-play and multi-agent improvement.** Multi-Agent Evolve (MAE) Chen et al. (2025b) co-evolves Proposer, Solver, and Judge roles using a single base model. The Judge is required because correctness is not directly verifiable in their domains. Countdown's rule-based verifier removes this need, decreasing the number of roles to two agents and making the Proposer's reward the central design choice. Multiagent finetuning Subramaniam et al. (2025) trains multiple specialist Solvers from one base model but does not learn the underlying data distribution.

**Curriculum learning for reasoning.** Self-Evolving Curriculum (SEC) Chen et al. (2025a) frames curriculum design as a multi-armed bandit over a *fixed* pool of problems, allocating compute by estimated learning gain. The "edge of learnability" framework Sundaram et al. (2026) argues that the most useful training examples lie at the boundary of solvability and proposes teacher-generated stepping-stone problems. We use the same principle in a fully verifiable setting and, unlike SEC, *generate* new problems rather than select from a fixed pool. This enables targeting harder difficulties but introduces a distribution-shift cost. The role of training-data composition in reasoning is also emphasized by DeepSeek-R1 DeepSeek-AI et al. (2025), which uses a manually curated mix.

## 3 Method

**Setup.** The Solver is Qwen2.5-0.5B initialized with SFT and fine-tuned with RLOO. The Proposer is a small MLP ($\sim$70k parameters) that, conditioned on a context vector $[\text{progress}, \text{recent pass rate}]$, outputs a factorized categorical distribution over three difficulty knobs: the number count $k$, the number magnitude, and the target magnitude. The knobs define a difficulty *region* and a rule-based generator samples instances from it and an oracle discards unsolvable ones with brute force. The Proposer chooses a difficulty region rather than concrete numbers to (i) reduce an intractable $\sim 10^{10}$ action space to a few dozen settings, making the sparse reward learnable, and (ii) keeps the Proposer non-LM, so any downstream effect is attributed to the curriculum and not to a second capable model.

**Proposer reward.** Let $p$ be the Solver's empirical pass rate on a proposed problem (fraction of 16 rollouts that are correct). The Proposer's reward is the information gain

$$r_{\text{IG}} = p\,(1-p), \tag{1}$$

which is also the variance of the Solver's Bernoulli success outcome. For RLOO with binary rewards, the variance of the advantage per-problem—and hence the policy-gradient magnitude—is proportional to $p(1-p)$. When the Solver always succeeds or always fails, all advantages are zero and no learning occurs, but at $p = 0.5$ the gradient is largest. Maximizing $r_{\text{IG}}$ therefore steers the Proposer to problems on which the Solver learns the most. We add a diversity bonus equal to the

Table 1: Countdown test performance on B1-B4(%). In-distribution (3–4 numbers, $N$=50) and out-of-distribution (5–6 numbers, $N$=256). Step counts: B1/B4 = 100, B2 = 85, B3 = 56 (some runs terminated early by Modal; evaluated at latest checkpoint).

| | In-distribution (3–4) | | | OOD (5–6) | | |
|---|---|---|---|---|---|---|
| Method | Avg | P@1 | P@16 | Avg | P@1 | P@16 |
| B1 Fixed | **59.0** | **55.7** | **74.0** | 6.2 | 0.3 | 2.3 |
| B2 Uniform | 49.4 | 46.1 | 74.0 | 6.2 | 1.3 | 6.6 |
| B3 Scheduled | 44.1 | 41.2 | 72.0 | 3.8 | 0.0 | 0.0 |
| B4 Learned | 40.5 | 39.5 | 66.0 | 1.6 | 0.0 | 0.0 |

policy entropy, $\lambda H(\pi)$, which equals $-\lambda \,\mathrm{KL}(\pi \,\|\, \mathrm{Unif})$ up to a constant and discourages the Proposer from collapsing onto a single difficulty. Here $\mathrm{Unif}$ means a uniform distribution over the Proposer's discrete difficulty knobs. The Proposer is trained with the same RLOO estimator as the Solver, using its batch of proposed problems as one group.

**Training loop.** Each step: (1) the Proposer samples difficulty knobs and a generator samples solvable problems, (2) the Solver produces 16 rollouts per problem which are scored by the verifier, (3) the binary pass rate $p$ gives the Proposer reward $p(1-p)$ and the Proposer is updated with RLOO + entropy bonus, (4) the same rollouts, with the full $\{0, 0.1, 1.0\}$ verifier reward, update the Solver via RLOO. Because the rollouts come from the Solver as it was at the start of the step, the Proposer is rewarded by a one-step-lagged Solver and both updates use a single sampling pass.

## 4 Experimental Setup

We compare four problem sources, all training the same Qwen2.5-0.5B Solver from the same SFT checkpoint with identical RLOO settings (16 rollouts per problem, $\sim$100 update steps): **B1 Fixed** (the curated `countdown_tasks_3to4` dataset), **B2 Uniform** (problems generated uniformly over the parameter ranges), **B3 Scheduled** (a hand-designed easy-to-hard curriculum in which number and target magnitude ramp up linearly and $k$ (number of input numbers) increases from 3 to 4 at the midpoint), and **B4 Learned** (our Proposer). Each checkpoint is evaluated on a held-in test set (3–4 input numbers, $N$=50) and a held-out OOD set (5–6 input numbers, $N$=256), sampling 16 responses per problem and reporting average accuracy and pass@$k$ (a response is correct iff the verifier returns 1.0). We then study two fixes in a $2 \times 2$ grid search: the diversity coefficient $\lambda \in \{0.1, 0.3\}$ and curated-data mixing $\rho \in \{0.0, 0.5\}$ (the fraction of each Solver batch drawn from the curated dataset, with the Proposer rewarded only on its own problems).

## 5 Results

### 5.1 Quantitative Evaluation

**The learned curriculum underperforms and degrades the Solver.** Table 1 reports in-distribution and OOD performance. For in-distribution, the ranking is B1 > B2 > B3 > B4. The fixed dataset (RLOO milestone) is best and the learned curriculum (our Proposer) is worst. All three generated-problem methods (B2–B4) trail the curated original dataset (B1). This is consistent with the distribution shift between generated and curated problems, and the Proposer trails even the simpler generated baselines.

Evaluating the B4 Solver at intermediate checkpoints reveals that in-distribution accuracy *falls monotonically* with training step (step 25: 48.9%, step 44: 47.6%, step 100: 40.5%) (i.e. more curriculum training makes the Solver worse).

### 5.2 Qualitative Analysis

**The cause is Proposer collapse.** Figure 2 shows the Proposer's policy entropy over training: it falls toward zero and flatlines for the final third of training. Because the diversity coefficient was too weak ($\lambda$=0.01), the information-gain reward drives the Proposer onto a single, hard difficulty setting. The

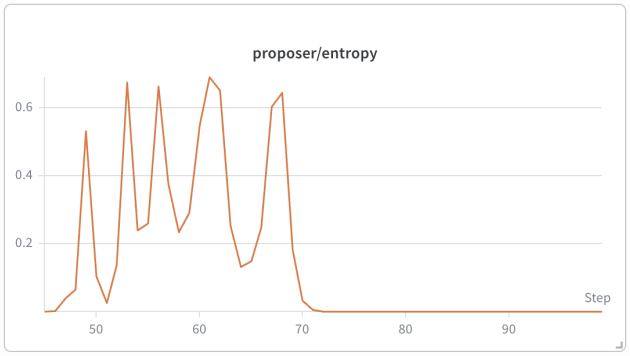


Figure 2: Proposer policy entropy collapses to zero (flatlining over the final third of training), concentrating the curriculum on a single difficulty.

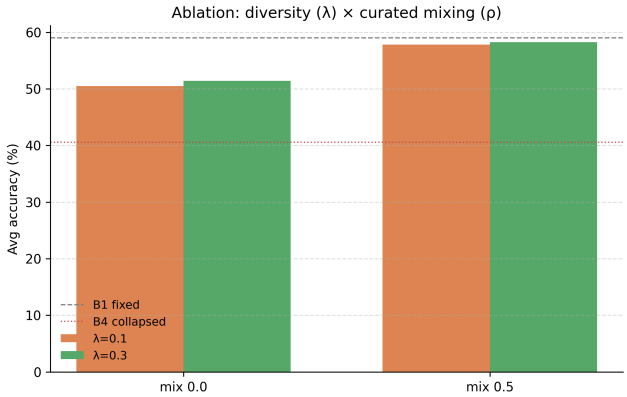


Figure 3: Diversity ($\lambda$) $\times$ curated-mixing ($\rho$) ablation on in-distribution average accuracy. Both no-mix bars are $\sim$10 points above the collapsed $\lambda$=0.01 run (dotted) and both mix-0.5 bars reach the fixed-data B1 baseline (dashed).

Solver then trains on an increasingly narrow band of problems and overfits, explaining the monotonic degradation. Thus, the diversity term is a critical stabilizer rather than an optional hyperparameter.

**Diversity and mixing fixes recover the curriculum.** The diagnosis suggests two independent fixes, which we test in a $2 \times 2$ grid search over the diversity coefficient $\lambda$ and curated-data mixing $\rho$ (Table 2, Figures 3 and 4). Both behave as predicted and help recovery. **Diversity prevents entropy collapse** by raising $\lambda$ from 0.01 to 0.1–0.3, which (no mixing) lifts average accuracy from $40.5\%$ to $\sim 51\%$, recovering to the level of B2, the uniform-generation baseline, ($49.4\%$). Within this range $\lambda$ is insensitive (50.5 vs 51.4, within noise at $N$=50). **Mixing closes the distribution gap** as adding $50\%$ curated data lifts accuracy even further to $57.8$–$58.3\%$, matching the B1 fixed-data RLOO milestone ($59.0\%$). The best config ($\lambda$=0.3, $\rho$=0.5) reaches $58.3\%$ average accuracy and $56.1\%$ pass@1—tied with B1—while exceeding it on average pass@$k$ (mean pass@$k$ $75.1\%$ vs $69.8\%$; Figure 4), potentially because curriculum-trained Solvers produce more diverse rollouts. We confirmed (via the `proposer/entropy` curves) that all four fixed-$\lambda$ runs avoid the entropy collapse of the $\lambda$=0.01 run. We emphasize that with $N$=50 the differences among the top configurations and B1 are within noise.

**OOD generalization is capability-bound**. On problems with 5–6 input numbers, all methods sit near the floor (average accuracy 1.6–6.2%, pass@1 $\approx$ 0; Table 1). The 3–4 $\rightarrow$ 5–6 capability gap exceeds what any training-distribution method achieves at 0.5B scale, so the OOD hypothesis cannot be tested here.

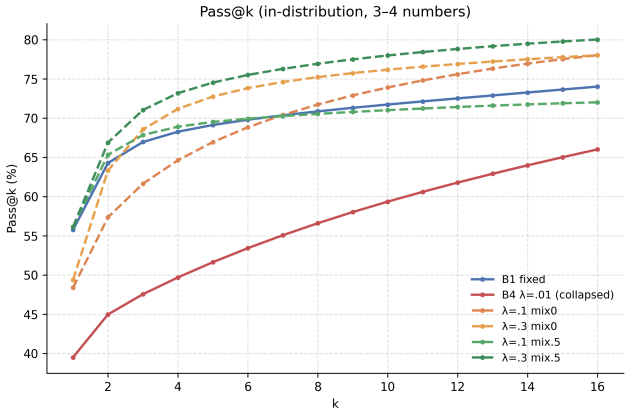


Figure 4: In-distribution pass@$k$. The collapsed curriculum is lowest at every $k$; all four fixed runs match or exceed B1, the fixed-data RLOO milestone baseline.

Table 2: Diversity ($\lambda$) and curated-mixing ($\rho$) ablation for in-distribution (%). All four fixed runs avoid the entropy collapse of the $\lambda$=0.01 run. ($N$=50, `fix_div0.3_mix0` evaluated at step 75 due to Modal failure.)

| $\lambda$ | $\rho$ | Avg Acc. | P@1 | P@16 |
|---|---|---|---|---|
| 0.1 | 0.0 | 50.5 | 48.4 | 78.0 |
| 0.3 | 0.0 | 51.4 | 49.4 | 78.0 |
| 0.1 | 0.5 | 57.8 | 56.1 | 72.0 |
| 0.3 | 0.5 | **58.3** | **56.1** | **80.0** |
| B1 Fixed (ref.) | | 59.0 | 55.8 | 74.0 |
| B4 ($\lambda$=0.01, collapsed) | | 40.5 | 39.5 | 66.0 |

## 6 Discussion

Our results present two findings. First, the information-gain objective is insufficient alone: rewarding $p(1-p)$ from a capable initialization drives the Proposer toward ever-harder problems and, without a strong diversity term, collapses the curriculum onto a single difficulty. This actively harms the Solver and degrades it as training goes on. The diversity coefficient is the crucial hyperparameter. Once it's large enough to prevent collapse, performance is insensitive to its exact value. Second, there is a tension between *generation* and *distribution match* (proportion of each in dataset). Reaching genuinely OOD difficulty requires generating new problems, but generation introduces a shift from the curated distribution (the gap between B2–B4 and B1 before mixing). Mixing curated data into a generative curriculum recovers the gap while retaining the ability to generate different difficulties. With both fixes, a self-generated curriculum matches a curated dataset—a result worth noting given that it creates its own problems rather than relying on human curation.

**Limitations.** Our evaluation sets are small ($N$=50 in-distribution), so differences among the best configs and the fixed-data B1 baseline are within noise. Thus, this is not a decisive win. Also, several runs terminated early due to Modal failures and are evaluated at slightly fewer steps. As for the OOD hypothesis, it remains untested at this scale.

## 7 Conclusion

We implemented and analyzed a verifier-driven learned curriculum for Countdown. The naive curriculum collapsed as Proposer's entropy fell to zero and flatlined, narrowing training onto a single difficulty and degrading the Solver. We diagnosed this mechanically and showed that two fixes recover it: a strong diversity regularizer to prevent collapse and mixing curated data to close the generation–curation distribution gap. Together, these match the fixed-data RLOO milestone. The takeaways are that self-generated information-gain curricula are viable but require strong diversity

regularization and that OOD generalization at this scale is bounded by model capability rather than the training distribution.

## 8 Team Contributions

- **Kai Wen:** All components—SFT/IPO/RLOO baselines, the Proposer architecture and information-gain reward, the generator and solvability oracle, curated-data mixing, the OOD evaluation set, all training and evaluation runs, and analysis and writing.

**Disclosure of AI Tool Use.** Consistent with the Default Project Guidelines permitting AI tool use on the project extension (and only the extension), AI tools were used to assist with implementing the extension code and drafting/editing this report. All ideas, experimental design, runs, and analysis are my own.

**Changes from Proposal** For compute efficiency on a single GPU, we adopted a one-step lag (a single sampling pass shared by both updates) instead of the proposal's proposed frozen reference Solver refreshed every $K$ steps. The OOD-generalization hypothesis proved untestable, so the contribution shifted toward an analysis of why the learned curriculum fails and how diversity and mixing affect it.

## 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:2402.14740 [cs.LG] https://arxiv.org/abs/2402.14740

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

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. 2025b. Multi-Agent Evolve: LLM Self-Improve through Co-evolution. arXiv:2510.23595 [cs.AI] https://arxiv.org/abs/2510.23595

DeepSeek-AI et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL]

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of Search (SoS): Learning to Search in Language. arXiv:2404.03683 [cs.LG] https://arxiv.org/abs/2404.03683

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent Finetuning: Self Improvement with Diverse Reasoning Chains. arXiv:2501.05707 [cs.CL] https://arxiv.org/abs/2501.05707

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] https://arxiv.org/abs/2601.18778