CS224R Milestone: IPO + RLOO Alignment Report

Matthew Torre[1] and Donnie Brooks Raymond[1]

[1]Stanford University, mtorre04@stanford.edu raymondd@stanford.edu

Abstract

Curriculum learning, where an agent sees progressively harder problems over time, is a simple and intuitive way to improve the reasoning ability of large language models (LLMs) trained with reinforcement learning. For this to work, the curriculum needs two things: a way to estimate problem difficulty and a way to choose problems that match the agent's current skill level. We propose an extension to RLOO (REINFORCE Leave-One-Out) that handles both pieces with one mechanism borrowed from competitive chess: an Elo rating system. Problems and the agent are each assigned a rating that is updated online from the binary correctness reward, and a Gaussian scheduler (with a small exploration probability) samples problems whose rating is near the agent's current rating. We evaluate the approach against an SFT baseline and a vanilla RLOO baseline under matched training budgets. The Elo-curriculum variant gets the strongest Pass@1 result, but the advantage does not carry over to larger sampling budgets, where vanilla RLOO matches or beats it. By logging the rating dynamics during training, we trace this gap to a cold-start problem: because every problem starts with the same rating, the scheduler ends up drawing too often from the small pool of problems whose ratings have actually changed. We discuss warm and heuristic rating initialization and forced early exploration as remedies.

1 Introduction

Reinforcement learning (RL) has become a standard tool for eliciting reasoning behavior from large language models, with verifiable-reward setups producing large gains on mathematics, code, and planning tasks [5]. One recurring issue in this setting is the reward signal itself. If the model almost never solves a problem, the reward is too sparse to give much useful gradient information. If it already solves the problem reliably, there is not much left to learn from that example.

Curriculum learning is a natural response to this problem. If the agent sees easier problems early and harder problems only as it improves, training can spend more of its budget on examples where the model is actually able to learn. Recent work has shown that these kinds of curricula can improve RL fine-tuning of LLMs. Building a curriculum, though, requires two key decisions. First, we need difficulty estimation: how should the relative difficulty of each problem be measured? Common approaches include hand-built heuristics, a pass over the data with a strong reference model, a learned teacher agent, or procedurally generated synthetic problems. Second, we need scheduling: once we have difficulty estimates, which problems should be sampled at each step? Gaussian and cosine schedules and teacher-driven selection have all been used.

We use an Elo rating system, the relative-ranking scheme popularized by chess, to combine these two decisions. Each problem and the agent itself carry a numerical rating; after every attempt, the ratings are updated from the binary outcome exactly as in a chess match, so that problems the agent solves lose rating and problems it fails gain rating. This lets us estimate difficulty online based on the model's current performance, and scheduling becomes a matter of sampling problems with ratings close to the agent's rating. We add this mechanism on top of RLOO without otherwise changing the training recipe.

Our contributions are:

- An Elo-based curriculum for RL fine-tuning of LLM reasoners that jointly handles difficulty estimation and scheduling, using a Gaussian sampler over agent-problem rating gaps with a small exploration probability.
- A controlled comparison against SFT and vanilla RLOO baselines under matched budgets, including Pass@k results and full traces of the rating dynamics.
- An analysis of why the curriculum underperforms vanilla RLOO in our setup, identifying uniform rating initialization as the likely cause, and concrete proposals (warm/heuristic initialization, forced early exploration) for closing the gap.

2 Related Works

2.1 Self-Evolving Curriculum for LLM Reasoning

Chen et al. frame curriculum design as a non-stationary multi-armed bandit problem that is solved at the same time as RL fine-tuning. Their method, SEC, learns a curriculum policy that selects problem categories based on immediate learning gain, using the absolute advantage from the policy-gradient update as a proxy, and updates this policy with a TD(0) rule. Across planning, inductive reasoning, and mathematics they report improved generalization to harder, out-of-distribution

problems and better skill balance when training on multiple domains at once. Our work has a similar goal of estimating difficulty online, but instead of learning a separate bandit policy, we use a simpler Elo update.

2.2 Curriculum Reinforcement Learning From Easy to Hard Tasks Improves LLM Reasoning

Parashar et al. introduce the E2H Reasoner, which schedules tasks from easy to hard during RL with verifiable rewards. One of their main findings is that easy problems help early in training, but they need to be faded out on the right schedule to avoid overfitting. They study probabilistic schedulers, including a Gaussian scheduler with a moving center, and also provide theoretical analysis in an approximate policy-iteration framework. Their approach is the closest to ours because both use a Gaussian scheduler. The main difference is that their center is manually controlled, while ours is determined by the agent's Elo rating.

2.3 DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-R1 [5] shows that large-scale RL with verifiable rewards can encourage sophisticated reasoning behaviors, including long reasoning traces, self-verification, and reflection, sometimes without elaborate reward shaping. This motivates studying whether choices about the training distribution, such as curriculum ordering, can make this kind of self-improvement more effective.

2.4 Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability

Sundaram et al. focus on the setting where RL stalls because the model's initial success rate is close to zero. Their SOAR framework uses a teacher copy of the model to propose synthetic stepping-stone problems for a student copy. The teacher is rewarded based on the student's improvement on a held-out set of hard problems, rather than an intrinsic proxy reward. They show models can escape reasoning plateaus on problems with 0/128 empirical success, and that the structural well-posedness of generated problems matters more than their solution correctness. This is a strong version of the teacher-agent approach to difficulty estimation. Our Elo scheme is a cheaper ratings-based alternative that does not require generating new problems.

2.5 Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs

Gandhi et al. study why some models self-improve under RL while others plateau. They use Countdown as a testbed and find that Qwen-2.5-3B improves much more than Llama-3.2-3B under the same training setup. They connect this gap to four behaviors: verification, backtracking, subgoal setting, and backward chaining. The stronger model already shows these behaviors, while the weaker model initially lacks them. They also show that priming the weaker model with these behaviors can unlock improvement. We use the same Countdown setting and binary verifiable reward, but focus on whether a curriculum can shape which problems the model sees during this self-improvement process.

3 Data

We train and evaluate on the Countdown arithmetic task: given a set of source numbers and a target, the model must combine the source numbers with arithmetic operations to reach the target. The task has an exact, automatically checkable answer, giving a binary reward $r \in \{0, 1\}$ where $r = 1$ indicates a correct solution. Our training set contains approximately 500,000 examples. Because the reward is verifiable and binary, it serves directly as the match outcome in the Elo update described in Section 4. The same structure also gives us simple difficulty heuristics: for example, the cardinality of the source-number set or the sum of all source numbers, which we return to when discussing rating initialization.

4 Methods

Base learner. Our base learner is RLOO (REINFORCE Leave-One-Out). We keep the RLOO objective, optimizer, and rollout procedure fixed and add only the curriculum machinery described below; the curriculum changes which problems are sampled at each step, not how the policy is updated from them.

Elo difficulty. We treat every training problem as a player in a rating system, and we treat the agent as one more player. Following the standard Elo formulation, a match between the agent (rating R_A) and a problem (rating R_P) produces scores

$$S_A = \mathbb{1}[r = 1.0] \in \{0, 1\}, \quad S_P = 1 - S_A, \tag{1}$$

In other words, the agent wins if and only if it solves the problem. Expected scores are given by the logistic Elo curve

$$E_A = \frac{1}{1 + 10^{(R_P - R_A)/400}}, \quad E_P = 1 - E_A, \tag{2}$$

and ratings are updated toward the realized outcome:

$$R'_A = R_A + K_A(S_A - E_A), \tag{3}$$

$$R'_P = R_P + K_P(S_P - E_P). \tag{4}$$

We use a per-problem step size $K_P = 800/N$, where N is the number of times that problem has been attempted, so that a problem's rating stabilizes as evidence about it accumulates, and a fixed agent step size $K_A = 32$. Ratings are initialized to $R_A = 1340$ for the agent and $R_P = 1500$ for every problem.

	SFT Baseline	RLOO Final	RLOO + Elo Curriculum
Pass@1	28.6	50.5	**56.1**
Pass@2	44.1	63.7	**64.5**
Pass@4	59.6	**71.5**	68.7
Pass@8	72.0	**75.9**	70.7
Pass@16	**78.0**	**78.0**	72.0

Table 1: Pass@k (%) on the held-out evaluation set. Best in each row is bolded.

Scheduler. At each step we sample problems with a Gaussian kernel centered on the agent's current rating:

$$S_{\text{Gaussian}}(t,k) = \exp\left(-\frac{(x_t-\mu_k)^2}{2\sigma^2}\right), \quad x_t = \text{agent Elo}, \tag{5}$$

where μ_k is the rating of candidate problem k and $\sigma = 200$. This concentrates sampling on problems near the agent's skill level, rather than problems that are either too easy or too hard, and the center moves upward automatically as the agent's rating rises. To avoid collapsing onto a narrow band of problems, with probability 5% we instead sample a problem uniformly at random (exploration).

Initialization note. We initially tried to set each problem's starting rating by running the SFT baseline over the training set and using empirical pass rate as a difficulty proxy. With approximately 500,000 examples, this was too expensive to run, so we used the uniform $R_P = 1500$ initialization above. As we discuss in Section 6, this choice turns out to matter.

5 Experiments/Results

Setup. We use the same training setup as our RLOO baseline: 100 training steps with a batch size of 128. The only difference between the baseline and our method is the Elo curriculum. To study what the curriculum is doing, we log the mean rating of sampled problems, the scheduler's target rating, and the agent's rating throughout training.

Final performance. Table 1 reports Pass@k for the SFT baseline, the final vanilla RLOO model, and our RLOO + Elo curriculum model. The Elo-curriculum variant gets the strongest Pass@1 (56.1, versus 50.5 for vanilla RLOO and 28.6 for SFT), suggesting that the curriculum helps the model commit to a correct single attempt more often. This advantage does not persist as the sampling budget grows: at Pass@4 and above, vanilla RLOO matches or exceeds the curriculum (e.g. Pass@16: 78.0 for RLOO versus 72.0 for the curriculum). Overall, though, the curriculum is worse than vanilla RLOO at larger values of k.

Training dynamics. Training rollout accuracy rises steadily over the 100 steps, while the RLOO loss stays noisy, which is expected for on-policy RL training.

Rating dynamics. The curriculum metadata shows the intended behavior at a high level: the scheduler's target rating climbs from the initial 1340 to roughly 1475, and the agent's own rating tracks it upward over the same range, while the mean rating of sampled problems hovers noisily around the 1400-1550 band. The gap between the smoothly rising target/agent rating and the much noisier sampled-rating mean is the first hint of the failure mode we describe below.

6 Conclusion

Our Elo-based curriculum over RLOO improved Pass@1, but it produced worse results than vanilla RLOO at larger sampling budgets. We think the main cause is a cold-start problem created by uniform rating initialization. Because every problem starts at $R_P = 1500$ and only the first few attempts on a problem move its rating by much, early in training almost all problems are indistinguishable to the scheduler. As a result, the Gaussian sampler ends up reusing the small set of problems whose ratings have already been updated, instead of exploring the full approximately 500,000-example pool. Our logging still shows that the number of unique problems seen grows roughly linearly with training steps, so this is not a total coverage failure. The issue is more that too much sampling mass is concentrated on the already-updated problems.

This suggests three concrete fixes that we would try next:

- **Warm rating initialization.** Initialize problem ratings from a pass over the data with the SFT model. This is expensive at full scale, but using approximately 1% of the dataset for both rating initialization and training could make it tractable.
- **Heuristic rating initialization.** Initialize ratings from a cheap task-specific heuristic, such as the number of source values or the sum of all source numbers, so the scheduler has a useful spread of ratings from step zero.
- **Forced early exploration.** Increase the exploration probability early in training, or schedule it from high to low, so the model quickly builds a larger pool of rated problems before switching to greedier Gaussian sampling.

All three changes target the same root cause: giving the scheduler a more informative rating landscape early in training. We expect this would preserve the Pass@1 gain without sacrificing as much performance at larger values of k.

7 Contributions

Both authors contributed to designing the Elo-based curriculum, implementing the method on top of the RLOO codebase, running experiments, analyzing results, and writing the report. Matthew Torre can be reached at `mtorre04@stanford.edu`. Donnie Brooks Raymond can be reached at `raymondd@stanford.edu`.

References

[1] Chen et al. Self-Evolving Curriculum for LLM Reasoning. 2025.

[2] Gandhi et al. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. 2025.

[3] Parashar et al. Curriculum Reinforcement Learning From Easy to Hard Tasks Improves LLM Reasoning. 2025.

[4] Sundaram et al. Teaching Models to Teach Themselves: Reasoning at the Edge of Learnability. 2025.

[5] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.