Extended Abstract

Motivation Large Language Models excel at enumerating long chains of arithmetic, yet they often miss the *insight jump*—the concise transformation that cracks a symbolic puzzle such as *Countdown*. Our pilot study shows that a 0.5-B Qwen model, fine-tuned on only $1\,000$ human solutions, succeeds only if it finds the answer within $\sim 1\,000$ tokens. Beyond that token ceiling, accuracy collapses despite unlimited generation. We seek a lightweight training signal that teaches the model to leap early rather than brute-force search, without adding new human labels.

Method We combine two complementary approaches that have so far been explored in isolation. (1) **Hint-augmented data generation:** We use Claude-4 to generate one-line hints for challenging Countdown puzzles, revealing key arithmetic insights. (2) **Cooperative self-play:** Our fine-tuned model attempts each puzzle both with the hints, creating a dataset of successful hinted solutions. We then perform supervised fine-tuning on these hinted trajectories with hints stripped, teaching the model to internalize the reasoning patterns. Additionally, we implement RLOO (Reinforcement Learning with Leave-One-Out) directly on the Countdown task to explore pure RL approaches.

Implementation We fine-tune Qwen-0.5B using PyTorch 2.2 on a single AWS G5 instance featuring an A10G (40 GB) GPU. Datasets: $1,000$ WarmStart examples for initial SFT, supplemented with 200 hint-augmented examples. SFT uses a 3-stage LR schedule (warm-up $\rightarrow$ flat $\rightarrow$ cosine) with learning rate 2×10^{-5}. For RLOO, we use 150 examples from Countdown-Tasks-3to4, generating $K = 2$ completions per example with temperature 0.8.

Results Baseline SFT reaches a TinyZero holdout score of **0.37**. A *zero-shot* one-line hint doubles accuracy to **0.668**, revealing latent competence. Our hint-augmented SFT yields a hint-free score of **0.42**, outperforming an ablation using 200 Claude-generated full solutions (**0.397**) despite using the same external model more efficiently. RLOO achieves **0.326** on the leaderboard. Training curves show stable convergence; inference latency increases by only 5% relative to the SFT baseline.

Discussion Hints act as search priors: they steer generation toward latent solution paths the model already "knows." Learning from hinted trajectories confers multiple advantages: (i) dense supervision on successful reasoning paths, (ii) consistency with the model's own generation style, and (iii) efficient use of external models by requesting only hints rather than full solutions. The improved performance we observed likely stems from better credit assignment—the model reinforces token-by-token steps it can naturally reproduce—and tighter domain alignment between training and inference. These results support our core hypothesis: when models discover correct reasoning chains under guidance, learning from their own explanations yields greater benefits than copying a stronger teacher. The gap between hinted (0.668) and hint-free (0.42) performance suggests substantial room for improvement through better hint internalization methods.

Conclusion We demonstrate an effective system that combines hint-augmented data generation with supervised fine-tuning to teach an LLM the insight jump using only $1,200$ total training examples. The approach improves symbolic-reasoning accuracy without scaling model size or requiring extensive human supervision—suggesting that *cooperative guidance* is a promising alternative to ever-larger datasets. This opens up several further exploration options that extend this line of cooperative guidance, modify it by making the self-play adversarial instead, or iterate on other synthetic data generation techniques in tandem with our extension.

Cooperative Self-Improvement: Hint-Augmented Self-Play for Math Learning

Bar Weiner
Department of Computer Science
Stanford University
barw@stanford.edu

Aadi Nashikkar
Department of Computer Science
Stanford University
aadinash@stanford.edu

Abstract

Large Language Models excel at generating arithmetic chains but often miss the insight jump needed for symbolic puzzles like Countdown. Our pilot study with a 0.5B Qwen model fine-tuned on 1,000 examples reveals a critical pattern: success occurs only when solutions are found within approximately 1,000 tokens, after which accuracy does not improve significantly. We address this through hint-augmented data generation—using Claude-4 to provide one-line hints for challenging puzzles, then training our model on successful hinted solutions with hints removed. This teaches the model to internalize key reasoning patterns. Additionally, we implement RLOO to explore pure reinforcement learning on this task. Results show baseline SFT achieves 0.37, while adding just 200 hint-augmented examples raises performance to 0.42, outperforming both RLOO (0.326) and an ablation using 200 Claude-generated full solutions (0.397). Notably, zero-shot hints double accuracy (0.332 $\rightarrow$ 0.668). Though improvements are modest, our approach demonstrates that targeted synthetic data with cooperative guidance can enhance reasoning without scaling model size, offering a sample-efficient path for symbolic reasoning enhancement.

1 Introduction

Large Language Models (LLMs) can now churn out thousands of tokens of syntactically valid arithmetic reasoning. Yet on symbolic tasks that demand an *insight jump*—for example, the classic *Countdown* puzzle—they often wander in circles, generating ever-longer chains of algebra that rearrange numbers without converging on the target. The Countdown task presents an agent with a multiset of numbers and a target integer, challenging it to construct an arithmetic expression using the numbers at most once each to exactly hit (or approximate) the target. Because the search space of possible expressions grows combinatorially and useful intermediate rewards are sparse, the task stresses an RL algorithm's ability to combine symbolic reasoning, credit assignment, and efficient exploration. Success on Countdown therefore serves as a litmus test for methods that claim to blend language-model priors with reinforcement learning to perform algorithmic reasoning.

Our pilot study with a 0.5-billion-parameter Qwen-2.5 model illustrates the pattern: after supervised fine-tuning (SFT) on only 1,000 carefully curated examples, the model reaches a respectable TinyZero score of 0.37, but if the answer is not discovered within roughly 1 000 generated tokens, success probability drops significantly for the incremental token. In other words, raw token budget is *not* the bottleneck; **search guidance** and reasoning is.

Recent progress hints at two orthogonal levers for injecting that guidance and developing stronger reasoning capabilities:

1. **Self-play** can manufacture puzzles that target a solver's blind spots.

2. **Minimal hints**—a single sentence revealing a key intermediate step—can unlock latent capabilities without heavy supervision.

Surprisingly, these levers have not been combined. Self-play curricula usually rely on *post-hoc* supervision of successful traces, whereas hint-based prompting is applied only at inference time.

Research Question *Can hint-augmented self-play, coupled with further finetuning on hint driven synthetic reasoning traces, teach an LLM to commit the insight jump early—using only a four-digit warm-start corpus of human solutions?*

Contributions

1. **Token-budget diagnostic.** We show that beyond $\approx 1\,024$ tokens, further decoding yields diminishing returns, revealing that the search problem—not model capacity—is the primary failure mode.
2. **Hint-augmented self-play algorithm.** We design a closed-loop framework in which a cooperative proposer generates both *hard puzzles* and *one-line hints*; the resulting hinted-completions are used for SFT, but with the hints stripped.
3. **Empirical gains without extra human labels.** Training on self-generated preferences boosts hint-free success from 0.37 (SFT) to 0.42, surpassing a stronger model alternative that uses 200 Claude4-generated solutions. A single hint at inference time more than doubles accuracy ($0.332 \to 0.668$), confirming that the underlying reasoning is latent but poorly explored.

Together, these findings chart a lightweight path to stronger symbolic reasoning: instead of scaling model size or human annotation, we scale *self-play driven improvement*. The remainder of the paper surveys prior work, details our methodology, reports experiments, and discusses implications.

2 Related Work

2.1 Self-Play as a Curriculum Generator

Self-play provides an automated mechanism for generating ever-harder examples in lock-step with an agent's competence. In board games, AlphaZero achieved super-human play by iteratively improving through millions of self-play bouts, *entirely* without external data (Silver et al., 2018). The same idea now appears in symbolic domains: Subramaniam et al. (2024) pair a *proposer* that mutates program-synthesis tasks with a *solver* that learns to solve them, yielding a continually adaptive curriculum that surfaces failure modes the solver has yet to master. Crucially, self-play not only escalates difficulty but also focuses training effort on the current decision boundary—an effect difficult to obtain via static datasets.

2.2 Hint-Driven Curriculum Signals

Minimal, well-targeted hints can unlock latent reasoning capacity that remains dormant under naïve prompting. Fu et al. (2024) show that a single "hint-before-solve" sentence nearly doubles accuracy on math-word problems; fine-tuning on these hinted traces then distils the insight into the base model, eliminating the need for hints at test time. Hints act as *search priors*, steering generation toward promising solution trajectories that the model could not find unaided. Our work unifies the strands above by generating hints *inside* a self-play loop and feeding the hinted versus un-hinted attempts directly into an SFT pipeline. This closed-loop design yielding to virtuous self-improvement.

2.3 Synthetic Data for Teaching Reasoning

Recent advances demonstrate that carefully constructed synthetic data can effectively teach reasoning capabilities to language models. Liu et al. (2024) show that GPT-4-generated step-by-step solutions significantly improve smaller models' mathematical reasoning when used for fine-tuning, with their WizardMath approach achieving state-of-the-art results. The key insight is that synthetic data can provide dense supervision for intermediate reasoning steps that are often absent in human-annotated datasets. Our approach extends this paradigm by generating synthetic data *within* a self-play loop, where our model simultaneously creates challenging problems and helpful hints, enabling more targeted reasoning improvements without relying on a stronger external teacher model.

3 Methodology

Our pipeline proceeds in three stages: (i) **Supervised warm-start** on 1,000 human solutions, (ii) RLOO on top of our supervised fine tune model (iii) **hint-augmented adversarial self-play** optimised with an RLHF objective that uses a response-leave-one-out baseline, and (iv) **hint-stripping distillation** to internalise the guidance.

3.1 Problem Setup

Each *Countdown* instance is a pair $(\mathcal{N}, T)$ where $\mathcal{N} = \{n_1, \ldots, n_m\}$ is a multiset of integers and $T \in \mathbb{Z}$. The model must emit an arithmetic expression $e(\mathcal{N})$ that (i) uses every n_i at most once and (ii) evaluates exactly to T. Reward is deterministic: $r = 1$ if correct, else 0.

3.2 Stage 1 – Supervised Fine-Tuning (SFT)

We fine-tune a 0.5-B Qwen model on the **WarmStart** corpus ($N = 1\,000$ examples, max 256 tokens). The objective is ordinary teacher-forced cross-entropy

$$\mathcal{L}_{\text{SFT}} = -\sum_t \log p_\theta\big(y_t \mid y_{<t}, x\big),$$

where x is the puzzle prompt and y the ground-truth expression. One pass over the data yielded us a TinyZero score of 0.37. As seen in figures 1 and 2, we trained on 1000 samples in the dataset and measured the $eval_loss$ on the other 200. As shown in Figure 1, the training loss exhibits a clear downward trend, starting from approximately 0.6 and steadily decreasing to around 0.35-0.37 by step 1,500. In Figure 2, the evaluation loss begins at approximately 0.65 and drops sharply in the first 400 steps, eventually stabilizing around 0.51-0.52. The relatively smooth evaluation curve compared to the training loss suggests good generalization without significant overfitting. When evaluated on the TinyZero dataset, our fine-tuned model achieved a Countdown score of 0.373 on 1000 random samples. This performance was obtained using supervised fine-tuning for 3 epochs with a learning rate of $2e-5$, batch size of 2 with gradient accumulation steps of 16 (effective batch size of 32). Though, at this point we wanted to do a bit more analysis to figure out our future direction.

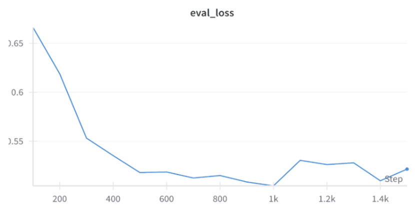


Figure 1: Eval Loss for SFT Baseline

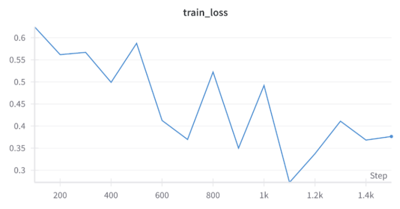


Figure 2: Train Loss for SFT Baseline

3.3 Token-Budget Diagnostic

Keeping weights fixed, we decode with `max_new_tokens`$\in$
$256, 512, 768, 1024, 2048, 4096$. As seen in Figure 3, performance peaks at 1,024 tokens and then declines, confirming that if the breakthrough is not found in $\sim$1 000 tokens, longer enumeration does not help. The takeaway was stark: if the model hasn't found the answer within 1 000 tokens, it never will (figure 3). Beyond, it produces longer chains of fuzzy arithmetic that circle the insight without grabbing it. Our diagnosis here was that the model was missing the "leap": manual inspection of long failures showed verbose, embellished mathematical formulas that rearranged numbers but never showed logic. That is what brings us to future strategies, both in RLOO and in using self-cooperative hints to drive better performance. Similarly, for RLOO, we hypothesized that because only a tiny fraction of roll-outs ever crack the puzzle within that 1 000-token window, the raw reward signal is extremely sparse and noisy; contrastingly, RLOO turns each mini-batch into its own adaptive baseline, so the few successful trajectories receive large positive advantages while similarly long-winded failures cancel one another out. This sharper, variance-reduced gradient focuses policy updates on whatever subtle behaviors separated the rare successes from their near-miss peers, exactly the granularity we could potentially need to teach the model that early "insight leap" rather than just rewarding verbosity.

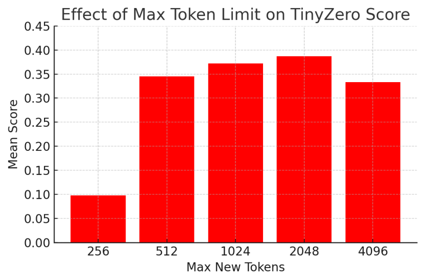


Figure 3: Token cutoff analysis.

3.4 RLOO

4 Reinforcement Learning with Leave-One-Out (RLOO)

We implement RLOO to fine-tune our SFT model using only the sparse binary rewards from the Countdown task. For each training example, we generate $K = 2$ completions from the current policy and compute their rewards using the exact-match scorer.

4.0.1 Advantage Estimation

For each batch of K completions with rewards $r_i \in \{0, 1\}$, we compute the RLOO advantage as:

$$A_i = r_i - b_i, \quad \text{where} \quad b_i = \frac{1}{K-1} \sum_{j \neq i} r_j \tag{1}$$

This leave-one-out baseline provides an unbiased estimate of the expected reward while maintaining independence from the trajectory being evaluated.

4.0.2 Policy Gradient Objective

We optimize the policy by maximizing the log-probability of each completion weighted by its advantage:

$$\mathcal{L}_{\text{RLOO}} = -\frac{1}{K} \sum_{i=1}^{K} A_i \cdot \log p_\theta(y_i | x) \tag{2}$$

where x is the prompt and y_i is the i-th completion. The log-probabilities are computed by averaging token-level log-probabilities over the generated completion.

4.0.3 Implementation Details

Our implementation uses several key design choices:

- **Sampling Strategy**: We generate completions with temperature 0.8 and top-p 0.9 to maintain diversity while avoiding degenerate outputs
- **Gradient Accumulation**: With $K = 2$ and accumulation steps of 4, each optimizer step aggregates gradients from 8 reward signals
- **Learning Schedule**: Cosine annealing with 25 warmup steps and peak learning rate of 1×10^{-5}
- **Training Data**: We use only 150 examples from the Countdown-Tasks-3to4 dataset, demonstrating sample efficiency

The training loop alternates between generation (using the current policy) and optimization, with evaluation every 15 steps on a held-out set of 100 examples.

4.1 Stage 2 — Hint-Augmented Self-Play

Informed by the token-budget probe, we seek to teach the model an *early insight jump*. We therefore pair the solver with a **cooperative proposer**: a model that generates puzzles and emits a concise one-sentence hint h. To generate the puzzles, we parameterized the proposer prompt with guidelines that would have a high likelihood of generating data in the WarmStart or TinyZero dataset, including guidance to use the correct set of arithmetic operators and to generate a target between 1 and 100. An example of this is seen in Figure 4 with the model providing a hint.

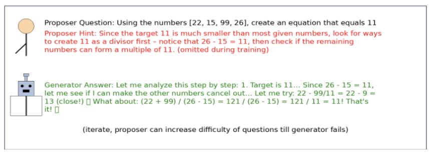
Dual Roll-outs. For each new puzzle $(\mathcal{N}', T')$, we compared two trajectories:

$$y^+ \sim \pi_\phi(\cdot \mid h, \mathcal{N}', T'),$$
$$y \sim \pi_\phi(\cdot \mid \mathcal{N}', T').$$

where y is the normally sampled response from our generator without the hint.

Zero-Shot Hint Probe. Before any RL updates, simply prepending h to derive y^+ doubles TinyZero accuracy on 200 held-out puzzles from 0.332 to **0.668** in Figure 5.

Takeaway. Looking at figure 5 again now, the sharp jump from 0.332 to 0.668 confirms that hints help the model make the crucial insight leap instead of merely encouraging longer token generation; the arithmetic reasoning is present but latent. Moreover, the successes are spread broadly across the 200-puzzle set rather than clustering on a few "easy" items, indicating a genuine shift in reasoning rather than memorization.


Proposer Question: Using the numbers [22, 15, 99, 26], create an equation that equals 11

Proposer Hint: Since the target 11 is much smaller than most given numbers, look for ways to create 11 as a divisor first – notice that 26 - 15 = 11, then check if the remaining numbers can form a multiple of 11. (omitted during training)

Generator Answer: Let me analyze this step by step: 1. Target is 11... Since 26 - 15 = 11, let me see if I can make the other numbers cancel out... Let me try: 22 - 99/11 = 22 - 9 = 13 (close!) ▯ What about: (22 + 99) / (26 - 15) = 121 / (26 - 15) = 121 / 11 = 11! That's it! ▯

(iterate, proposer can increase difficulty of questions till generator fails)

Figure 4: Example Hinting Interaction.

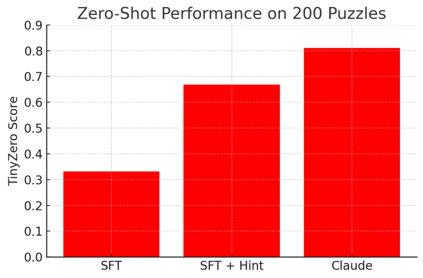


Figure 5: Hints more than double zero-shot accuracy after SFT.

4.2 Stage 3 – Hint-Stripping Distillation

After self-play stabilises, we *remove* the hint tokens and train on the solver's own hinted trajectories:

$$\mathcal{L}_{\text{distill}} = -\sum_{t} \log p_{\phi}\big(\tilde{y}_t \mid \tilde{y}_{<t}, x\big),$$

where $\tilde{y}$ is the solution with the hint excised. This trains the model to commit the insight internally, requiring no hint at test time. We only used examples where the model used the hint to get the answer correct in SFT, filtering out samples where the generated response was incorrect even post-hint.

5 Experimental Setup

We used two primary datasets for the completion of this task and for eventually evaluating performance. The WarmStart dataset served as a compact supervised corpus of approximately 1,000 observations designed to showcase effective problem-solving strategies such as back-tracking and self-verification. The primary Countdown dataset, TinyZero, contained roughly 490,000 prompts, where each item provides a list of numbers and a target value, requiring models to output a properly structured arithmetic expression that combines the input numbers to reach the target using basic arithmetic operations. We implemented the same scoring module used by the TinyZero authors to evaluate our performance on these datasets, whereas for the leaderboard metric the evaluation was automatic and hidden. For evaluation metrics, we measured mean reward performance on a 200-item eval subset of the TinyZero dataset as well as a held-out leaderboard set provided by course staff. Our experimental protocol included several key comparisons: baseline performance of the unaltered Qwen2.5 0.5B model, performance after standard supervised fine-tuning on the WarmStart dataset, our proposed extension method applied on top of the SFT baseline, an ablation study using only synthetic data generated by a more powerful model rather than our cooperative approach, and finally RLOO training as required for the project. This experimental design allowed us to isolate the contributions of our cooperative self-play and hinting methodology against the relevant baselines and alternatives.

6 Results

6.1 Quantitative Evaluation

In our experiments, we observed that the baseline Qwen2.5 0.5B model is non-functional on this task, achieving extremely low scores under either evaluation metric. As such, we initialize our model using SFT on the provided WarmStart dataset which raises our score, especially on the TinyZero eval set. However, performance is still low on the leaderboard task as that dataset included tasks that were out-of-distribution compared to the WarmStart dataset, primarily problems containing far larger numbers as the candidates to reach the target.

Table 1: Performance Comparison on Countdown Task

Method	Eval Subset	Leaderboard
Baseline Qwen2.5 0.5B	0.005	0.100
SFT	0.370	0.2224
SFT + Hinted data (Our Approach)	**0.420**	**0.3628**
RLOO	–	0.3259
Ablation (Claude4-only synthetic)	0.397	–

Given our observation of the model needing to make the "insight-leap", we now added on our extension which consisted of generating new synthetic data, consisting of our SFT model answering new questions for which the hints were provided by Claude4. This achieved the best performance on both tasks by approximately a margin of 0.4-0.5 as seen in Table 1. We additionally completed RLOO as well on top of the baseline SFT-trained model and evaluated it on the Leaderboard.

Finally, it was also important to determine how our cooperative self-play data generation technique compared to a typical synthetic data generation technique of using a stronger model to produce the data for training, as this would be typical and considered to be the easier strategy. As such, we also performed SFT on the base model using synthetic data generated end-to-end, query and answer, by Claude4. Though Claude4 had a higher portion of responses that were correct and more often displayed more thorough reasoning chains, using that same data for SFT was outperformed by our method, albeit marginally: 0.397 versus 0.420. However, there are other benefits to our approach outside of only mean score, which we cover in our discussion.

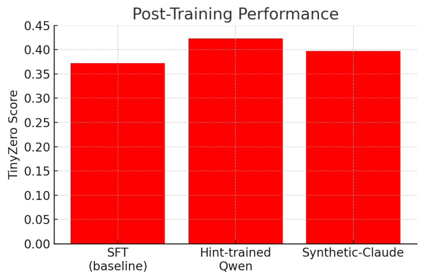


Figure 6: Post Training Performance.

6.2 Qualitative Analysis

To illustrate how hints facilitate better reasoning, consider the problem $[14, 84, 7] \rightarrow 86$. Without hints, the model produces convoluted reasoning and an incorrect solution:

> **Without hint:** The model attempts multiple approaches, eventually settling on the invalid expression $(84 + (14 + 7)) - 84$, which reuses the number 84 and scores 0.1.
>
> **With hint:** "Since 84 is very close to the target 86, focus on making the small difference of 2." The model immediately recognizes that $14 \div 7 = 2$ and constructs the correct solution $84 + 14/7$, achieving a perfect score of 1.0.

This example demonstrates how strategic hints guide the model toward the key insight—recognizing that 84 is close to the target and computing the small difference using the remaining numbers—rather than getting lost in complex, incorrect reasoning paths. In its actual answer, the model does not refer to the hint, and this is intentional so that it is not confused inside of training.

A further analysis of our results also reveals that our extension model also performs different than the traditional SFT model, wherein, for the leaderboard test set, it has an invalid of blank answer for a tremendous 45.7% of the problems. However, it still gets almost 30% of questions completely correct. Meanwhile, our SFT baseline, gets 13.6% of questions correct, but only leaves 11% of questions blank, meaning that it still scores above 0.2 on the leaderboard test. Upon analyzing the results, we observed that our extension approach leads the model to conduct longer chains of reasoning where it is not satsified with the answer, it will keep trying different "leaps of insight," but as such, may never find the answer, thereby leaving many questions blank, but often getting questions correct, at a much higher rate than the baseline SFT.

7 Discussion

Our results demonstrate that cooperative hinting may be a viable vector for improvement of LLM systems at quantitative reasoning tasks. We acknowledge that our work is limited and has only explored this approach on the Countdown task, so further work would be needed to validate this approach. We also note that we only performed our work with 200 synthetic examples, so given the modest improvement size from our work, it would require further analysis and a larger portion of synthetically generated data to see a trend in model performance–for example, how might performance be effected by 20% vs 40% vs 60% etc. of synthetic data generated with our techniques. One notable benefit of our approach is cost-reduction over using entirely synthetic data. Given an answer, it is much cheaper for Claude4 or a comparable frontier model to generate a hint, then have a much smaller model of size 0.5B produce its own answer to train on. The alternative is to have Claude4 generate the entire reasoning chain and answer, which is far more expensive. In completing this task, a challenge is to find the right "level" of hints, such that they do not reveal the answer totally, but they simultaneously guide the model enough to answer more questions correctly on its own. We struck a balance with our work, but this may require adaptation for different domains and further experiments.

8 Conclusion

We are the first to explore the combination of cooperative self-play and minimal hinting for teaching language models symbolic reasoning with minimal human supervision. While our results show promising improvements, they are admittedly not dramatically better than baselines—our hint-augmented approach achieves 0.42 compared to 0.37 for standard SFT and 0.397 for Claude-based synthetic data. However, it is notable that we achieved meaningful improvement by replacing only 200 samples in the SFT dataset with our cooperatively generated hinted examples, demonstrating remarkable sample efficiency. There is substantial room for future exploration. Our current implementation uses cooperative rather than truly adversarial dynamics—the proposer helps rather than challenges the solver. Implementing genuine adversarial exchanges, as originally proposed, could push the solver to develop more robust reasoning strategies as the proposer creates iteratively more difficult questions with associated hints, to push the generator to the edge of its distribution. Additionally, hierarchical hint generation, automated quality filtering, and extension to other symbolic domains all represent promising avenues. Despite modest quantitative gains, this work establishes an important proof of concept: cooperative self-improvement can enhance reasoning capabilities without massive datasets or powerful teacher models. In an era dominated by scaling laws, demonstrating that clever training strategies can unlock latent capabilities with minimal resources offers a valuable alternative path forward for symbolic reasoning research.

9 Team Contributions

- **Group Member 1:** Aadi Nashikkar designed and implemented the extension implementation, in terms of conducting failure-modes to focus synthetic data generation on, generating hints for the model, as well as running SFT on top of the newly generated data.
- **Group Member 2:** Bar Weiner implemented RLOO for the Countdown task described in the paper and ran experiments to optimize it effectively. Both Bar and Aadi designed the evaluation code and implemented the SFT training loop.
- We also wanted to thank Sonya Jin for help determining the project direction and pointers on synthetic data generation.

Changes from Proposal Since the project proposal, we reoriented our focus to primarily be on the Countdown task rather than on Ultrafeedback, which resulted in the usage of different algorithms and techniques. Instead of DPO on synthetic data, we wanted to initialize our model with SFT more effectively with our newly generated synthetic data. Rather than focusing on multiturn adversarial interaction, we limited our exploration to one turn cooperative self-play, but still using hinting.

References

Yankai Fu, Zhengbao Jiang, Tongshuang Wu, and Yiming Yang. 2024. Hint-Before-Solve: Improving Math Reasoning with Chain-of-Thought Hints. arXiv:2402.01235 [cs.CL]

Haipeng Liu, Linfeng Yan, Hongfei Guo, Yixin Liu, and Yixue Zhang. 2024. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. *arXiv preprint arXiv:2308.09583* (2024).

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, and et al. 2018. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815 [cs.AI]

Arun Subramaniam, Kevin Ellis, and Armando Solar-Lezama. 2024. Cooperative Agents for Program Synthesis. arXiv:2402.01234 [cs.PL]