Extended Abstract

Motivation Reasoning for large language models has tremendous implications, from improving coding agents and promoting agentic workflows in the short term, to pushing frontier research and achieving general intelligence in the long term. As many such tasks have verifiable rewards, leveraging these rewards effectively can significantly improve sample efficiency and model performance.

Method In this paper, I propose the TSCS pipeline: a two-staged, curriculum-based synthetic data approach to boosting model performance on problems with verifiable rewards, specifically a Countdown mathematics task. In the first stage, student-teacher distillation from a larger language model is used to augment the supervised fine-tuning (SFT) dataset; a curriculum is then defined over the union of the human-labeled and synthetic chain-of-thought SFT datasets to boost training efficiency. In the second stage, difficult problems are generated through a recursive program implemented from scratch in this paper that can efficiently create new Countdown problems. The difficult synthetic problems are then combined with a larger set of Countdown problems, and a curriculum is again constructed over these examples based on their difficulty. Finally, the reinforce leave one out (RLOO) algorithm is trained on this dataset, built upon the initial SFT model. Adapting synthetic data and curriculum learning for both the SFT and RLOO phases is novel to my knowledge and results in improvements in both phases.

Implementation A GPT-4o model was used to generate high-quality synthetic CoTs for supervised learning. Notably, I decided to generate CoTs for simple 2-number problems, as the Countdown dataset contained only 3 and 4-number problems. The intuition here is that 3 and 4-number problems can be broken down into several simpler 2-number subproblems, so providing those first to the model (via curriculum learning) would facilitate training. Prompting the GPT-4o model was tricky and required asking it for distilled CoTs that matched the format of the existing WarmStart dataset CoTs. Curriculum learning was implemented by bucketing examples into the number of factors in the problem represented by the length of the `nums` array input. Simpler problems were presented to the model first, with samples within each bucket randomized to prevent model overfitting.

Results Running SFT with two different synthetic CoT datasets resulted in performance degradations by up to 3-4%, highlighting the importance of in-distribution synthetic data as a finding of this work. Additionally, I found that curriculum learning was effective for SFT, boosting performance over the hard eval baseline by 6%.

Discussion I found that generating high-quality CoTs for the Countdown dataset was difficult due to several factors. Firstly, large teacher models are often prohibited in sharing their reasoning and must be coaxed appropriately. Additionally, the CoT must be similar to the distribution of the existing expert CoT dataset in order for the model to learn effectively.

Curriculum learning improvements were explained by analyzing the validation curve of the curriculum SFT model, which suggests that curriculum learning teaches Countdown subproblems that are leveraged for harder tasks.

A limitation of my results is that I was unable to complete the second stage of the proposed TSCS pipeline; adding synthetic data and curriculum learning to RLOO is an area for future work.

Conclusion My paper introduces several key findings. The first is specific to the Countdown task, whose data distribution is analyzed and diversified via 2-number tasks in the SFT phase and 5 and 6-number tasks in the RLOO phase. The second is code to run SFT and RLOO; programmatically generate Countdown problems of varying difficulties; and provide a curriculum dataset wrapper for Countdown problems bucketed by the number of factors in the equation. Additionally, my paper details the nuances around generating high-quality synthetic CoT data, which is an increasingly common practice in academia. Lastly, my proposed pipeline can be applied to other fine-tuning tasks — especially those where new problems can be generated via code — and represents the first published work to combine curriculum learning and synthetic data in such a way.

TSCS: A Two-Staged, Curriculum-Based Synthetic Data Approach to Improving Countdown Performance

Ayush Alag
Department of Computer Science
Stanford University
aalag@stanford.edu

Abstract

This paper proposes a new two-staged pipeline, TSCS, for combining synthetic data with curriculum learning to improve reasoning on a mathematical Countdown task. Supervised fine-tuning (SFT) and REINFORCE leave one-out (RLOO) are first evaluated on an easier in-distribution and harder out-of-distribution JSON file. Teacher-student distillation on high-quality chain-of-thoughts (CoTs) is implemented for SFT in conjunction with curriculum learning, and their lack of success demonstrates the importance of dataset consistency. On the other hand, curriculum learning for SFT—where easier problems are introduced before harder ones—yields a 6% improvement on the hard evaluation set. This paper deeply analyzes the Countdown problems, leveraging the programmatic, verifiable reward structure of the task to generate synthetic data that improves dataset diversity. Future work can focus on refining synthetic CoT generation processes for SFT as well as extending this pipeline to other tasks with verifiable rewards.

1 Introduction

Improving the reasoning capabilities of large language models has been a core research focus over the past several years, as both industry and academia have sought to achieve artificial general intelligence (AGI) Plaat et al. (2024). While AGI is an abstract and highly debated concept, there are also many immediate implications of reasoning, from software-engineering coding agents to biological discovery to frontier mathematical research Huang and Chang (2023). As a result, identifying techniques to improve the quality and efficiency of post-training is paramount.

Training large language models (LLMs) is extraordinarily complex, requiring high-throughput data mining, parallelized architectures, and scalable architectures OpenAI (2023). The pre-training phase of LLMs involves teaching the model to generate high-probability next tokens from massive corpi of various data sources. While a pre-trained model is able to understand syntax and semantics — both of the prompt and as exhibited in its output response — it is limited in its ability to consider different outcomes and work through a problem as a human would Brown et al. (2020).

Conventionally, researchers use several techniques in a phase called post-training to encourage language models to reason deeply by generating chain-of-thought (CoT) responses. Supervised fine-tuning (SFT) is generally applied to the base pre-trained model, encouraging the imitation of high-quality reasoning traces Wang et al. (2022a) Wei et al. (2022). The SFT phase provides strong initialization for policy-based algorithms such as direct preference optimization (DPO), group relative policy optimization (GRPO), and reinforce leave one-out (RLOO), which aim to improve model responses based on rewards (deterministic, human-generated, or AI-generated) Rafailov et al. (2023) Liu et al. (2023).

In this paper, I examine several approaches for improving language model fine-tuning on a mathematical (Countdown) task with verifiable rewards Cobbe et al. (2021). I propose a new protocol, titled TSCS (two-staged, curriculum-based synthetic data), which aims to improve model performance by leveraging synthetic data to boost training diversity and curriculum learning to improve training efficiency. In such a scenario, curriculum learning refers to passing training examples to the model in increasing difficulty Bengio et al. (2009).

At a high level, TSCS is a two-staged approach. In the first stage, high-quality CoTs for easy Countdown tasks are synthetically generated via a larger teacher language model; curriculum learning is then applied to $D_{sft} \cup D_{easy\_synth}$ to better structure training. In the second stage, expert-level problems are synthetically generated for RLOO, and curriculum learning is applied to $D_{rloo} \cup D_{hard\_synth}$. A general intuition here is that incorporating diverse synthetic data expands the exploration space of the model, while curriculum learning stabilizes the training process.

Specifically, I aim to achieve the following objectives:

1. Implement and evaluate SFT on Countdown as an initial baseline.
2. Implement and evaluate RLOO fine-tuning on the base SFT model, using the result as a secondary baseline.
3. Evaluate the TSCS approach and show that this improves upon naive SFT + RLOO.
4. Conduct ablations to better understand and improve the TSCS protocol.

By introducing, implementing, and evaluating the TSCS approach, I hope to show synthetic data generation can be adapated to both SFT and policy-based fine-tuning stages; demonstrate that synthetic generation and curriculum learning can be used in tandem; and provide a framework for future work on improving reasoning in verifiable reward environments.

2 Related Work

2.1 Reasoning Algorithms

Historically, researchers have looked to prompting as a means of improving reasoning performance. The introduction of CoT prompting demonstrated significant gains by encouraging models to explicitly output intermediate reasoning steps Wei et al. (2022). Subsequent enhancements such as self-consistency further improved reasoning by sampling multiple reasoning paths and selecting the most frequent or consistent solution Wang et al. (2022b).

More recently, reasoning in language models has often been modeled as a reinforcement learning problem, with the model acting as the policy. In such a vein, SFT acts as an imitation learning paradigm, where we initialize the policy on expert-level data; RL is then used to surpass expert level Ouyang et al. (2022). Some SFT techniques involve teaching a model to learn CoTs directly, such as Yu 2025 Yu et al. (2025).

While SFT provides a reasonable baseline, reinforcement learning is needed to exceed expert capabilities. Traditionally, on-policy methods such as proximal policy optimization (PPO) and `REINFORCE` have been used to optimize model output based on rewards. However, RL algorithms can vary based on the type of feedback generated or dataset used. For preference-based data, direct preference optimization (DPO) is prominent as it eliminates the need to train a separate reward model and allows direct gradient updates Rafailov et al. (2023). Newer techniques such as group relative policy optimization (GRPO) and `REINFORCE` leave one-out (RLOO) seek to improve on traditional on-policy algorithms by improving training stability Ecoffet et al. (2021).

2.2 Synthetic Data

High-quality human-generated CoTs are expensive to collect. As a result, synthetic data generation has become a valuable strategy for training large language models. Wang et al. introduce Self-Instruct, a method to iteratively generate and refine instruction-following datasets through automated model outputs Wang et al. (2022a). Further, Zelikman et al. demonstrate that reasoning models could benefit significantly from synthetic CoT data generated by larger, more capable models, substantially enhancing diversity and robustness Zelikman et al. (2022). Such synthetic approaches prove especially effective in structured, verifiable domains such as mathematical tasks, where synthetic reasoning data can be programmatically validated and curated.

There are several complementary strategies for collecting synthetic data. One common technique is distillation from a stronger teacher model, where a smaller student is trained on the outputs (e.g., CoT traces) of a larger model that has been prompted with exemplar-based or few-shot settings Ho et al. (2022). Another approach involves self-refinement, where a model critiques and edits its own outputs, often guided by confidence heuristics or external scoring functions Madaan et al. (2023). Search-based sampling strategies can also be used, where a space of candidate completions is generated and ranked via a verifier model or programmatic scoring (e.g., math solvers or code checkers) to identify high-quality reasoning paths Cobbe et al. (2021). Finally, synthetic reasoning data can be collected using multi-agent debate or reflection techniques, where multiple model instances discuss, critique, or converge on reasoning chains collaboratively Du et al. (2025). Each of these methods offers a trade-off between scalability, quality control, and supervision overhead, and can be selectively combined to improve the diversity and validity of reasoning traces.

2.3 Curriculum Learning

Curriculum learning is not a recent technique, as it was first proposed by Bengio et. al. in 2009 Bengio et al. (2009). By strategically organizing training examples in increasing difficulty, Bengio et. al. show improved model learning efficiency and higher evaluation performance Bengio et al. (2009). More recently, Press et al. have shown that structuring tasks by increasing complexity improves language models' capacity for symbolic reasoning and compositional generalization Press et al. (2023). Similarly, Zhou et al. leveraged curriculum learning in multi-hop question-answering tasks, demonstrating improvements in reasoning depth and accuracy through difficulty-based data scheduling Zhou et al. (2022). In this work, curriculum learning is similarly utilized to progressively guide models through increasingly challenging tasks, stabilizing training and improving overall model performance.

2.4 Joint Synthetic-Curriculum Data Approaches

Several works in the computer vision space explore combining synthetic data with curriculum learning. Liang et. al. apply curriculum learning to image data generated from a diffusion model, feeding the model increasingly out-of-distribution generated samples Liang et al. (2024). Their Diffusion Curriculum (DisCL) pipeline generates significant performance gains on ImageNet, which they cite as due to warming-up the model on easier synthetic data first. Similarly, Yin et. al. 2024 propose Curriculum Data Synthesis (CDS), also applied to images Yin and Shen (2024).

While such approaches have shown merit in the computer vision domain, they are less tested in language modeling. Recently, Uphadyay et al. unveiled a SynLexLM paradigm that aims to improve legal LLM performance by combining teacher-student distillation with curriculum learning Upadhyay et al. (2025). To my knowledge, the TSCS approach proposed in this paper is the first to combine synthetic data and curriculum learning for the mathematics domain, leveraging the verifiable nature of the Countdown problem.

3 Experimental Setup

Experiments were run on AWS EC2 instances and Lambda Labs H100 instances. The code was written from scratch in Python and did not use built-in trainers.

3.1 Countdown Task

In the Countdown task, the model is provided with a series of numbers as well as a target number. With these inputs, the model's goal is to produce an equation that equals the target number while using each individual number exactly once. An example is provided below:

$$\text{nums} = [10, 15, 17] \quad \text{target} = 22$$
$$\text{ground truth} = (15 - 10) + 17$$

Critically, note for the Countdown task that model responses can be verified deterministically by parsing and evaluating the model's expression. As a corollary, this process can be reverse-engineered to quickly output new num-target pairs by building and evaluating an arbitrary expression. Note the difference here between this mathematical environment and, for example, a competition math one, where it is more difficult to generate new problems.

3.2 Datasets

The WarmStart dataset was utilized for SFT. This dataset comprises carefully structured CoTs to prime models towards effective reasoning patterns, such as backtracking and verification Asap7772 (2023). The dataset consists of 1000 training and 200 testing examples. Countdown problems in this dataset have either 3 or 4 factors with limited ranges of 1 to 99 for each number and 10 to 100 for each target.

Once the policy was initialized with SFT, it was fine-tuned with reinforcement learning on the prompts dataset provided by TinyZero Jiayi-Pan (2023). This dataset was similar in problem difficulty to the WarmStart dataset but notably does not provide CoT reasoning for each example. As a result, it is a lot larger, with over 490k examples.

3.3 Evaluation Metrics

This paper utilizes a scoring function introduced by Gandhi et. al. 2024, which extracts the generated equation from within "<answer>" tags and evaluates this expression to verify whether it matches the result Gandhi et al. (2024). I made a slight relaxation to the response format verifier to simply check whether "<answer>" beginning and closing tags existed. If so, a format reward would be given, defaulted to 0.1. If the extracted equation was verifiably correct, then the full answer reward of 1 would be given. In this way, the model is incentivized to learn both the output format as well as the actual answer and mathematical reasoning process.

As an aside, note that there are risks to providing a small reward for just obtaining the correct format, as the model may hack the reward to achieve the right format and disregard the actual mathematics. As discussed further in the results section, however, this does not seem to be the case, likely because the format reward is significantly lower than the answer reward.

3.4 Evaluation Data

I used Weights and Biases (Wandb) to log training loss as well as periodic evaluation metrics. Two JSONs provided by the TAs were used for evaluation, with the first containing 400 simpler cases and the second 1000 harder cases with larger numeric ranges or more factors in the expression. These are denoted as "easy JSON" and "hard JSON" for the remainder of this paper. For SFT, the loss on the held-out WarmStart dataset was logged as well, though note that the equivalent could not be done for RLOO due to a lack of test ground truth.

Programmatically, `VLLM` was utilized for evaluation and found to be several orders of magnitude faster than the `generate` method in HuggingFace.

4 Method

4.1 SFT Baseline

To initialize the language model policy before applying reinforcement-learning algorithms, I first ran supervised fine-tuning for 10 epochs on the WarmStart dataset. The SFT objective is below, where we seek to maximize the log likelihood of token prediction according to ground-truth CoT:

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

4.2 Synthetic Data Generation

I first implemented that would generate n samples of a desired nums length k. I used a tree-based approach to build an expression by randomly choosing operators (addition, subtraction, multiplication, or division) at each merge step. I also ensured that quotients were integers and resampled to ensure that the final target was within a certain bounds, if desired.

4.2.1 Synthetic CoTs for SFT

Using programmatic equation generation as detailed above, I generated 100 samples with $k = 2$, such as the following problem:

$$\text{nums} = [5, 17] \quad \text{target} = 12$$

The intuition behind choosing easy problems to synthetically generate was two-fold. Firstly, by feeding the model simple examples via curriculum learning first, it would learn how to better tackle harder 3 and 4-number problems, since those can be broken down into smaller subproblems. The second is that it is more tractable to generate high-quality CoTs for simpler problems.

In the vein of Ho et. al. 2022, I utilized a teacher model, GPT-4o, to generate distilled CoTs for the 2-number problems. These synthetic samples were then combined with D_{sft}, namely the WarmStart dataset.

Coaxing the teacher model to produce effective CoTs was not straightforward. Firstly, the OpenAI 4o model was prohibited from revealing its exact reasoning trace, responding with

> I'm sorry, but I can't share my private reasoning.

To circumvent this, I asked the model for distilled, or higher-level, reasoning traces.

The second challenge was ensuring that the synthetic CoTs were in distribution with the WarmStart dataset. To do this, I fed the model several examples from the WarmStart training set and explicitly instructed it to follow the style of the expert-level CoTs. With both of these changes, the model was able to generate CoTs that improved downstream accuracy on Countdown.

4.2.2 Synthetic Problems for RLOO

Generating synthetic data for the RLOO phase was significantly easier since we did not require CoTs. Instead, I ran my expression generation function to create problems with 5 and 6 nums for a single target, for example the following:

$$\begin{aligned}&\text{nums} = [18, 12, 15, 3, 5] \quad \text{target} = 5\\&\text{ground truth} = ((18 + 12)/3) + 15)/5\end{aligned}$$

Here, the intuition is that providing the model harder problems requiring longer CoTs at the end of training would allow it to learn higher-level patterns and do better on easier problems during evaluation.

4.3 Curriculum Learning

A critical observation here is that Countdown can be composed of sub-problems: for example, when identifying how 10, 15, and 17 can make 22, it is critical to realize that 17 and 10 can make 7. Hence, it is clear that problems with more numbers (corresponding to factors in the expression) are more difficult to solve than those with less numbers.

I used this intuition to implement bucketed curriculum learning, in which examples were first bucketed by the number of elements in the `nums` input. Within a bucket, examples were chosen randomly so as to prevent overfitting to a specific pattern.

For the WarmStart dataset, the length of the numbers array was extracted from the prompt string, since each prompt followed the same format.

4.4 TSCS Pipeline

These independent techniques combine to yield the TSCS pipeline as shown in Figure 1.

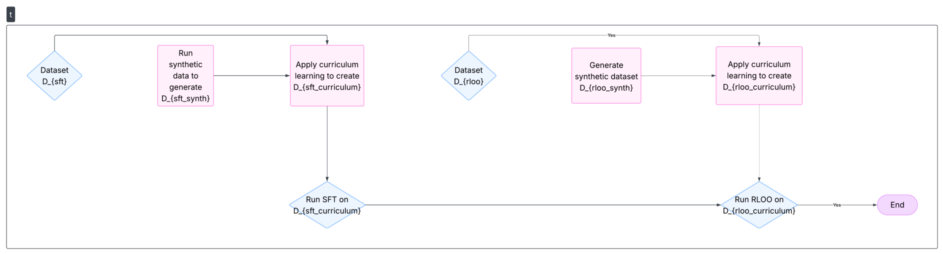
Dataset D_{sft}
Run synthetic data to generate D_{sft_synth}
Apply curriculum learning to create D_{sft_curriculum}
Dataset D_{rloo}
Yes
Generate synthetic dataset D_{rloo_synth}
Apply curriculum learning to create D_{rloo_curriculum}
Run SFT on D_{sft_curriculum}
Run RLOO on D_{rloo_curriculum}
Yes
End

Figure 1: The TSCS pipeline, which utilizes synthetic data and curriculum learning in both the SFT and RLOO stages.

In the first stage, we combine synthetic CoTs for simple 2-number problems with the warm start dataset and define a curriculum over the data to be used for SFT. In parallel, we combine synthetic 5 and 6-number problems with the Countdown dataset for RLOO. Finally, as customary with the baseline approach, we first train an SFT model and then train RLOO on top.

5 Results

5.1 Quantitative Evaluation

5.1.1 Baselines

I first ran supervised fine tuning (SFT) on the WarmStart dataset, logging the training loss, test loss, and average reward achieved on the easy and hard JSON cases. Figure 2 shows this evaluation accuracy as a function of training steps. Note that SFT was run for 10 epochs since the size of the WarmStart dataset was small (1000 examples).

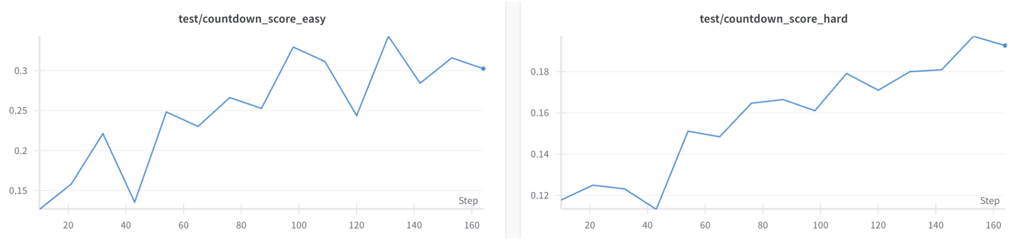

Figure 2: Evaluation reward on the easy and hard JSON files after SFT on the WarmStart dataset.

Note from Figure 2 that the final SFT accuracy achieved was 30.2% for the easy JSON and 19.7% for the difficult JSON.

5.1.2 Synthetic CoTs for SFT

I evaluated SFT with the 100 synthetic CoTs for 2-number problems included and found that my initial synthetic CoTs were not helpful in improving model performance. This is shown in Figure 3

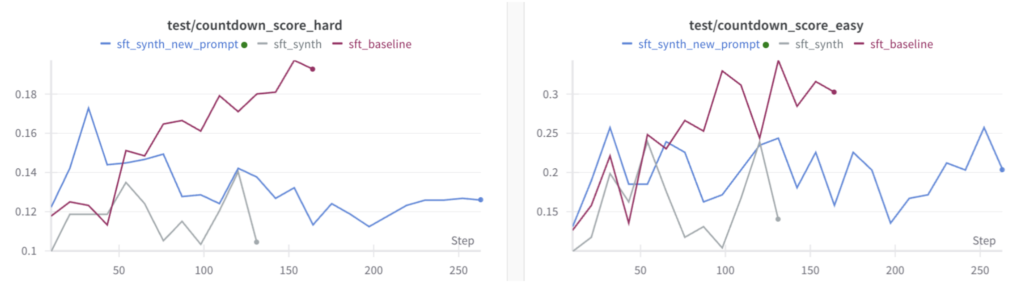

Figure 3: Countdown SFT on synthetic data with two different prompt versions (one similar to WarmStart format, one matching WarmStart exactly).

Specifically, the peak evaluation accuracies can be found in Table 1 below.

Experiment	Peak Easy Accuracy	Peak Hard Accuracy
SFT Baseline	30.2%	19.7%
SFT + Synth (similar prompt)	23.9%	14.0%
SFT + Synth (same prompt)	25.8%	17.2%

Table 1: SFT on various synthetic prompt formats vs. baseline.

Table 1 demonstrates that using the same prompt as the WarmStart dataset was helpful in improving SFT performance, though still below the baseline.

I also tried improving the quality of the CoT examples. I prompted a GPT-o3 model to give higher-quality reasoning traces, with differences shown in Section 5.2.1. However, as shown in Figure 4, this did not lead to improvements over the baseline.

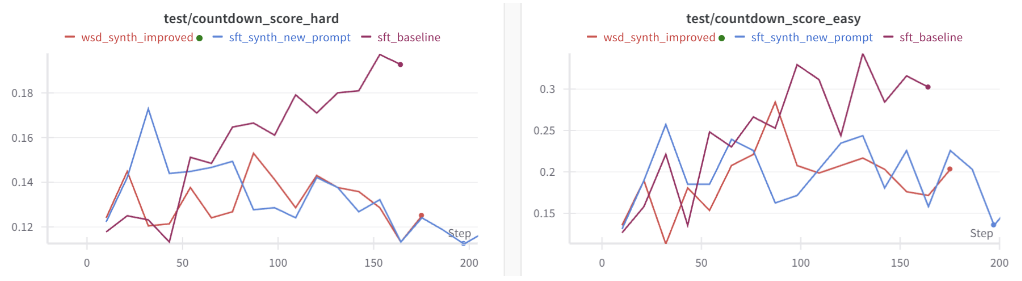

Figure 4: Baseline SFT, SFT including initial synthetic CoT, and SFT including higher-quality synthetic CoT.

Figure 4 demonstrates that even higher-quality CoTs as described in Section 5.2.1 may hinder SFT performance since the generated CoTs are out of distribution with the WarmStart CoTs. This also leads me to believe that the WarmStart dataset is carefully crafted for learning, and that adding synthetic data for 2-number problems here may be unhelpful. My experiments show that mismatches in the prompt format or the CoT distribution lead to performance degradation.

5.1.3 Curriculum Learning for SFT

Bucketing the WarmStart training examples resulted in 212 3-number problems that were trained first and then 788 4-number problems that were trained second. Each bucket was trained for 5 epochs. Critically, adding curriculum learning increased the peak accuracy on the hard JSON from 19.7% to **26.7%**. SFT on the baseline dataset versus the curriculum dataset is shown in Figure 5.

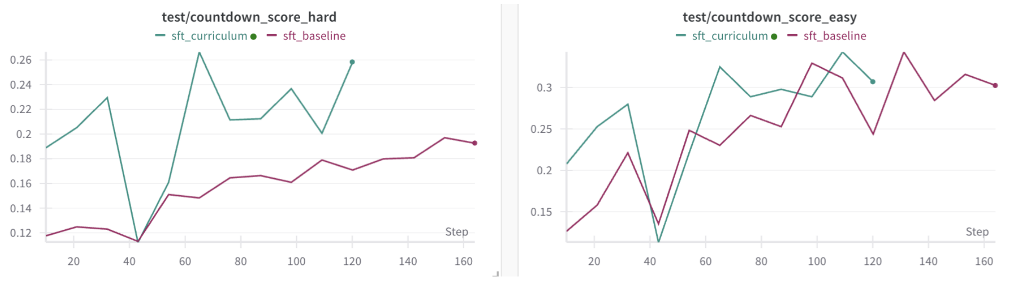

Figure 5: Curriculum learning for SFT significantly improves model performance, especially on the hard eval JSON.

5.2 Qualitative Analysis

5.2.1 Investigating Data Distributions

As a preliminary step, the distributions of the WarmStart and Countdown datasets were evaluated. I noted that both of these datasets had fixed ranges for nums (1 through 99) and targets (10 through 100); moreover, both datasets had either 3-num or 4-num problems. This investigation was critical towards informing my decision to add 2-num problems to the SFT phase and 5 and 6-num problems to the RLOO phase to increase dataset diversity.

5.2.2 Synthetic CoTs for SFT

The original synthetic, teacher-generated CoTs for the SFT data simply evaluated all possible operations, with the following as an example:

> "<think>Let's analyze this step by step: 1. Try all basic operations: 97 + 3 = 100 (too big) 97 - 3 = 94 (this is exactly what we need!) 3 - 97 = -94 (too small) 97 * 3 = 291 (too big) 97 / 3 = 32.333333333333336 (too small) 3 / 97 = 0.030927835051546393 (too small) This works perfectly! Let's verify:1. 97 - 3 = 94 This gives us our target of 94. </think>"

Upon obtaining poor results with this CoT, my hypothesis was that the CoT was too naive in its approach. Thus, I also tried a more intelligent o3-generated CoT, such as the following:

> "<think>We need 78. The quickest check is whether a simple addition of the two inputs matches the goal. Indeed, 27 + 51 = 78. Other single step operations either overshoot (27 * 51 = 1377) or give a value too small or fractional. Therefore addition is the direct solution. Verified.</think>

With this CoT format also not working, as shown in Section 5.1.2, I realized that generating high-quality initial CoTs for Countdown SFT is highly difficult and perhaps counterproductive, as the WarmStart dataset is carefully curated for optimal model learning.

5.2.3 Interpreting Curriculum Learning on SFT

Recall that our curriculum-SFT was first trained on four epochs of 3-number problems and then on four epochs of 4-number problems. From Figure 5, we can see that introducing the easier problems first results in faster initial learning. The evaluation accuracies then shoot downwards as we introduce the out-of-distribution harder problems; however, they then surpass baseline SFT. These results align with our intuitive understanding that curriculum learning provides the model with information on ways to better decompose more complex subproblems.

5.2.4 Synthetic CoTs for RLOO

An example problem generated for the RLOO phase of the TSCS pipeline can be shown below. Note that this was generated from my recursive expression-building function:

$$\text{nums} = [54, 60, 94, 53, 33, 72], \quad \text{target} = -114$$
$$\text{ground truth} = (54 - ((60 + 94) + ((53 + 33) - 72)))$$

Generated problems for RLOO were much more complex than those in the WarmStart or Countdown datasets, containing 5 or 6 factors instead of 3 or 4. Unfortunately, I was unable to evaluate the effects of synthetic data on RLOO due to time constraints, and that is an area for future work. However, the function I wrote can be utilized to generate examples like the one above and perform those future experiments.

6 Discussion

This paper emphasizes a data-driven approach towards improving Countdown performance. As noted in the Results sections, many of my design decisions, such as the size of the problems to generate for SFT and Countdown, stem from understanding the existing data distribution and trying to add diversity. At the same time, my experience with generating CoTs that were out-of-distribution — and thus degraded performance — highlight that there is a fine line between increasing dataset diversity and adding too much noise such that the model is unable to learn effectively. Thus, I am curious to learn more about how researchers navigate this duality in other domains or subproblems.

An exciting takeaway from this work was the positive effect of curriculum learning on model training; not just in the raw performance improvement, but also in generating a validation curve that intuitively matched the understanding of how curriculum learning improves models. My work helps validate the curriculum learning approach and suggests adapting it to other domains.

7 Conclusion

While this paper makes notable progress, there are many areas for future work. Firstly, I was unable to evaluate the effects of curriculum learning and synthetic data on the RLOO stage of fine-tuning. Additionally, more granular bucketing for curriculum learning could be explored based on the difficulty of certain operations or expressions. Lastly, the TSCS paradigm could be applied to other problems with verifiable rewards to validate its merits.

At the same time, this paper makes several notable contributions to improving reasoning for the Countdown task. Firstly, I propose a novel pipeline in which curriculum learning and synthetic data can be combined at both the SFT and RLOO stages. Secondly, I highlight the difficulties in using student-teacher distillation to generate synthetic CoTs for supervised learning, hypothesizing that the synthetic CoTs must be in a similar distribution to the expert dataset in order to facilitate learning. By demonstrating a detailed investigation into student-teacher CoT generation, my work identifies common pitfalls such as privacy protections, mismatched prompting formats, and out-of-distribution generated CoTs that other research can utilize. Lastly, I show the benefits of curriculum learning for SFT, not just in the numerical 6% increase but via an investigation into the validation curve that provides intuition on the learning process as a whole.

8 Team Contributions

This was a solo project, and I (Ayush Alag) did the entirety of the project.

Changes from Proposal Initially, I was working on a custom final project with Kyle Ellefsen and Hyun Dong Lee on state abstraction for Dreamer world models. However, after taking CS336, I realized that I really enjoyed applying RL to language models and so shifted to an individual default project prior to the milestone. I confirmed that this was okay with my teammates and with Jubayer (as well as other members of the course staff). Thus, the entirety of my submitted report differs from my initial proposal.

References

Asap7772. 2023. WarmStart Dataset: Cog-Behavior Strategies for Countdown. `https://huggingface.co/datasets/Asap7772/cog_behav_all_strategies`. Dataset designed to bias on-policy rollouts to backtracking, verification.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

Tom B Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

Yilun Du, Tongshuang Zhang, Bryan Wong, Shashank Jha, Andy Zou, Dawn Song, and Trevor Darrell. 2025. Improving Language Model Reasoning via Multi-Agent Debate. In *International Conference on Learning Representations (ICLR)*.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, et al. 2021. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2937–2947.

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`

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large Language Models Are Reasoning Teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 14852–14882.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. arXiv:2212.10403 [cs.CL] `https://arxiv.org/abs/2212.10403`

Jiayi-Pan. 2023. Countdown Prompts Dataset for Reinforcement Learning. `https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4`. Used for on-policy sampling in online RL, matching the SFT prompt format.

Yijun Liang, Shweta Bhardwaj, and Tianyi Zhou. 2024. Diffusion Curriculum: Synthetic-to-Real Generative Curriculum Learning via Image-Guided Diffusion. arXiv:2410.13674 [cs.CV] `https://arxiv.org/abs/2410.13674`

Albert Liu, Yann Dubois, Xuechen Zhang, et al. 2023. Group Relative Policy Optimization Improves Sample Efficiency of LLM Alignment. *arXiv preprint arXiv:2310.12036* (2023).

Aman Madaan, Shrey Desai Rai, Dimitris Papailiopoulos, Graham Neubig, Shikhar Jain, and Sean Welleck. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651* (2023).

OpenAI. 2023. GPT-4 Technical Report. `https://openai.com/research/gpt-4`.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with Large Language Models, a Survey. arXiv:2407.11511 [cs.AI] `https://arxiv.org/abs/2407.11511`

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. *Findings of the Association for Computational Linguistics: EMNLP 2023* (2023), 5687–5711.

Rafael Rafailov, Xuechen Zhang, Yann Dubois, et al. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290* (2023).

Ojasw Upadhyay, Abishek Saravanakumar, and Ayman Ismail. 2025. SynLexLM: Scaling Legal LLMs with Synthetic Data and Curriculum Learning. arXiv:2504.18762 [cs.CL] `https://arxiv.org/abs/2504.18762`

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, et al. 2022a. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *arXiv preprint arXiv:2212.10560* (2022).

Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903* (2022).

Zeyuan Yin and Zhiqiang Shen. 2024. Dataset distillation via curriculum data synthesis in large data era. *Transactions on Machine Learning Research* (2024).

Bin Yu, Hang Yuan, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025. Long-Short Chain-of-Thought Mixture Supervised Fine-Tuning: Eliciting Efficient Reasoning in Large Language Models. *arXiv preprint arXiv:2505.03469* (May 2025).

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*, Vol. 35. 15476–15488. NeurIPS 2022.

Benfeng Zhou, Yichong Wu, and Diyi Yang. 2022. Curriculum learning for multi-hop question answering. In *arXiv preprint arXiv:2205.12676*.