

Extended Abstract

Motivation The goal of our project was to explore supervised finetuning (SFT), reinforcement learning (RLOO), curriculum learning, and sampling techniques for training language models on reasoning tasks.

Method and Implementation We first post-trained Qwen/Qwen2.5-0.5B with SFT on the Countdown task. We then explored three approaches to further improve performance beyond the vanilla SFT model: 1) a hyperparameter search over generation parameters, 2) a curriculum learning approach for SFT, and 3) post-training the SFT model with reinforcement learning. Our hyperparameter search explored the influence of the number of beams in beam search and model temperature on generation quality. Our curriculum learning strategy sorts examples in the SFT dataset based on either completion length or answer length, then trains on shorter examples before longer examples. Our reinforcement learning strategy used the REINFORCE Leave One-Out (RLOO) objective to further train the model based on reward signals.

Results The results of our hyperparameter search showed that increasing the number of beams in beam search and using a high model temperature significantly improved generation results. The results of our curriculum learning strategy showed that, due to quirks in the SFT dataset, a version of curriculum learning where we *reversed* the curriculum actually outperformed normal curriculum learning and vanilla SFT. The results of our RLOO implementation did not improve upon vanilla SFT.

Discussion and Conclusion Taken together, our results highlight the importance of carefully constructing datasets for post-training LLMs. We are not sure why our RLOO implementation did not work, and any comments from the TA after looking at the code would be greatly appreciated.

Fine-Tuning Language Models with Curriculum Learning

Ethan Trepka

Department of Computer Science
Stanford University
trepka@stanford.edu

Abstract

Curriculum learning is an approach to train machine learning models by increasing the difficulty of training examples over the course of training. Here, we explored whether a curriculum learning strategy could improve performance of supervised fine-tuning of the language model Qwen/Qwen2.5-0.5B on the Countdown mathematical reasoning task. Due to quirks of the provided SFT dataset, we found that a version of curriculum learning where we reversed the curriculum and trained the model with more challenging examples before simpler examples actually outperformed normal curriculum learning and vanilla SFT. Taken together, this result highlights the importance of carefully constructing and filtering datasets for post training LLMs.

1 Introduction

Humans learn new topics, such as math skills, from carefully constructed curriculum. It is challenging to teach someone calculus before teaching them basic arithmetic. Similarly, machine learning models oftentimes benefit by learning simpler concepts and tasks before more complex ones. This approach is known as ‘curriculum learning’ in the machine learning literature (Bengio et al. (2009)).

Curriculum learning can be divided into data-centric curriculum learning, where the complexity of the training data/task is increased over the course of training and model-centric curriculum learning, where the task remains the same but the model capacity increases over the course of training (Soviany et al. (2021)).

In data-centric curriculum learning problems, there are two key components: the *curriculum scheduler* which determines when to increase the difficulty of training examples and the *difficulty criterion/curriculum selection method* which determines how to increase the difficulty of training examples over the course of training.

Here, we explore whether curriculum learning can be used to improve the performance of supervised fine-tuning of a Qwen model on the countdown task. We address the curriculum scheduling and selection problems for our SFT dataset by sorting training examples either by completion length or answer length, then training on examples of increasing length. Due to some quirks of the provided SFT dataset, we found that the strategy where we reversed the curriculum actually outperformed normal curriculum learning. Separately, we also explored a variety of RL and sampling techniques to further improve the performance of our base SFT model.

2 Related Work

Bengio et al. (2009) introduced the notion of ‘curriculum learning’ - where models are trained with increasingly challenging examples - to the machine learning literature. They applied curriculum

learning to simple tasks, such as classifying a geometric shape in an image, and illustrated that curriculum learning improves both the speed of convergence and quality of the minima identified in these tasks. Soviany et al. (2021) review the use of curriculum learning approaches in machine learning, and present a unified framework of curriculum learning methods. The review encompasses both data-level curriculum learning, as in Bengio et al. (2009), and model-level curriculum learning where model-capacity increases over training time. Our exploration of curriculum learning here falls under data-level curriculum learning, as we gradually increase the complexity of the training data over the course of training.

Gandhi et al. (2025) apply a carefully designed curriculum learning approach for post training LLMs with reinforcement learning. They first observed that different base LLMs, such as Llama-3.2-3B and Qwen-2.5-3B differ substantially in their performance after RL post-training, with Qwen outperforming Llama because the base Qwen model exhibited superior initial reasoning behaviors. This deficit could be resolved by applying a 'curriculum learning' style strategy to post train Llama - first by teaching it basic reasoning strategies, then by post training it on a specific task. Our exploration of curriculum learning here differs from Gandhi et al. (2025) because we do not try to 'teach reasoning strategies' in a separate step, then post train on a different step. Instead, we hope that the model first acquires simple reasoning strategies then more complex reasoning strategies over the course of training on increasingly complex data for a single task, Countdown.

3 Method and Experimental Setup

We first post-trained Qwen/Qwen2.5-0.5B with SFT, followed by RLOO on the Countdown task as specified in the default project handout. More details on specific parameters for these runs are provided in the results section.

We then explored two extensions to the base default project. In the first extension, we explore how different sampling strategies affect the performance of Qwen/Qwen2.5-0.5B post trained with SFT on the Countdown task. More specifically, we test beam search with different numbers of beams, different temperatures, and different top-p. For this extension, we report performance (combined format score and correctness score) on the leaderboard.

In the second extension, we explore how curriculum learning affects the performance of Qwen/Qwen2.5-0.5B post trained with SFT on the Countdown task. We sort the training examples in the SFT dataset based on either the length of the entire completion or the length of the answer string ("answer complexity"). In each epoch, we then train the model on the sorted data either in order of shortest to longest or longest to shortest ("reverse"). We report performance (format score, correctness score, combined score) on a held-out evaluation set that is sampled IID from the dataset. In Figure 1, we show the first and last examples from the dataset when sorted by completion length and by answer length.

4 Results

Our Qwen/Qwen2.5-0.5B model post trained with SFT achieved a combined score of 0.415 on the held-out examples on Math Reasoning Leaderboard 1 (Table 1). The performance of the model over training epochs is visualized in Figure 2. This run used a learning rate of $5e-5$, batch size of 2 with 8 gradient accumulation steps (effective batch size of 16), 1024 token max completion length, and early stopping with a patience of 2 epochs.

	Math Reasoning Leaderboard 1	Math Reasoning Leaderboard 2
SFT	0.415	—
RL	—	0.280
Extension	—	0.3016

Table 1: Performance on Math Reasoning Leaderboards for Milestone (Leaderboard 1) and Final Submission (Leaderboard 2)

Curriculum by Answer Complexity

Curriculum by Completion Length

Length Curriculum: Shortest Completion First

Prompt: [15, 20, 4] \rightarrow 39
Completion: <think> Let me try to find a way to reach 39 using these numbers.
 First attempt:
 $15 + 20 = 35$, then $35 + 4 = 39$
 This works! Let me verify backwards:
 $39 - 4 = 35$
 $35 = 15 + 20$
 So we can reach 39 by first adding 15 and 20 to get 35, then adding 4 to get 39.
 </think> <answer> $(15 + 20) + 4$ </answer>

Length Curriculum: Longest Completion Last

Prompt: [80, 6, 50, 2] \rightarrow 61
Completion: <think> Let me try to work towards 61 systematically:
 1. First, let's try to get close to 61 with the larger numbers 80 and 50:
 $80 - 50 = 30$, still need to get to 61
 $80 + 50 = 130$, too large
 2. Let's try division with the larger numbers:
 $80 / 2 = 40$, this could be useful as it's closer to 61
 $50 / 2 = 25$, also closer to target
 3. Let's work with $80/2 = 40$:
 $40 + 6 = 46$
 $40 + 50 = 90$, too large
 4. Let's try a different approach with 80:
 $80 - 50 = 30$
 $30 * 2 = 60$
 $60 + 6 = 66$, getting closer but still not exact
 5. Let's try one more sequence:
 $80 / 2 = 40$
 $40 + 50 = 90$
 $90 / 6 = 15$, too small
 ...
 ...
 ...
 </think>
 <answer> Unable to find a valid solution to reach 61
 </answer>

Answer Complexity: Shortest Answer First

Prompt: [37, 9, 24, 41] \rightarrow 15
Completion: <think> Let me try to find a way to get to 15 using these numbers.
 First, let me look for factors or multiples of 15 that might help:
 $15 = 3 * 5$
 Since we have 24, dividing by something close to 1.5 would get us closer to 15.
 Let me try working with 24 first:
 $24/9 = 8/3$ (not helpful since we need integers)
 $24+9 = 33$ (too far from 15)
 $24-9 = 15$! This is our target!
 Now I need to verify if I can use 37 and 41 in a way that doesn't affect this result.
 $37-41 = -4$
 $37+41 = 78$
 ...
 ...
 ...
 </think>
 <answer> $24 - 9$ </answer>

Answer Complexity: Longest Answer Last

Prompt: [85, 67, 38, 54] \rightarrow 50
Completion: <think> Let me try to find a way to get to 50 using these numbers.
 First, let me try to get closer to 50 by subtracting larger numbers.
 $85 - 67 = 18$ (this gets us a smaller number to work with)
 But 18 is too small to reach 50.
 Let me try dividing larger numbers:
 $85 / 38 \approx 2.2$ (not helpful as it's not an integer)
 Let me try another approach:
 $54 - 38 = 16$
 $67 - 16 = 51$
 This gets us close to 50!
 ...
 ...
 ...
 </think> <answer> $(54 - 38) = 16$; $(67 - 16) = 51$; $(51 - 85) = -34$; $(-34 + 84) = 50$ </answer>

Figure 1: Examples of Two Curriculum Learning Strategies for the Countdown Task

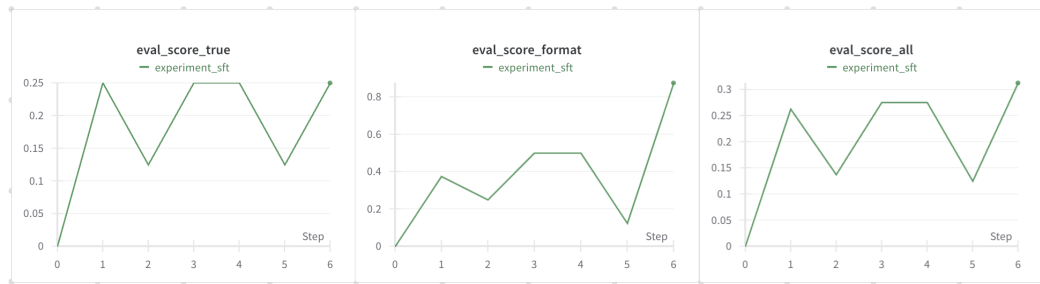


Figure 2: Visualization of countdown correctness score, format score and combined score over the course of training with SFT for multiple epochs.

Our subsequent RLOO experiments did not results in a substantial performance improvement of the Qwen/Qwen2.5-0.5B model post trained with SFT, with a performance on Math Reasoning

Leaderboard 2 of 0.28. We explored many different hyperparameters and extensively proofread the code, but were unable to pinpoint why this was the case. We show example results from runs with learning rates of $1e-6$ or $5e-6$, k values of 8 or 12 (see definition of RLOO objective in project handout), and where we only trained on samples where at least one generation out of k had the answer correct ("skiperr") or where we also include samples where at least one generation of out k merely had the format correct ("all"). These results are illustrated in Figure 3. The run names in the legend follow the conventions described above.

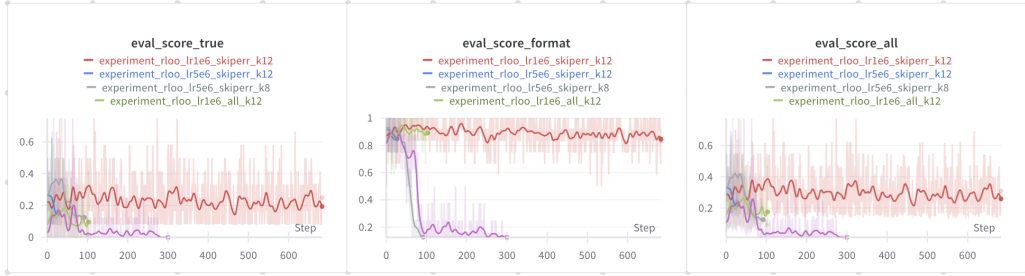


Figure 3: Visualization of countdown correctness score, format score and combined score over the course of training with RL on up to 600 samples.

We then explore a simple extension to the baseline SFT model where we tested a variety of different sampling strategies, including beam search with 3 or 4 beams and different values of temperature and top-p. Contrary to the parameters recommended on Ed by the teaching staff (lower temperatures, top ps), we found that a high temperature of 1, top p of 1 and using 4 beams for beam search substantially improved results (Table 2). Surprisingly, this was sufficient for our extension model to pass the 0.3 threshold on Math Reasoning Leaderboard 2 (Table 1).

Table 2: Performance of different sampling strategies for countdown task.

Run Name	Sampling Strategy	Score
Beam-4 (Temp=1, Top-p=1)	Beam search, 4 beams	0.3016
Beam-3 (Temp=1, Top-p=1)	Beam search, 3 beams	0.2926
Beam-3 (Temp=0.5, Top-p=0.8)	Beam search, 3 beams	0.2602

Finally, we explored whether curriculum learning could improve performance when post training with SFT. Due to compute limitations, we only trained these models for two epochs. Over the two epochs, we did not see a substantial benefit of either the 'completion length' or 'answer complexity' curriculum learning strategies. However, when we inverted these strategies by training on the longer/more complex examples *first* rather than last, we actually saw a performance improvement over vanilla SFT when training on longer completions before shorter completions (Figure 4; blue line in evaluation-score-all plot higher than green line after both epochs).

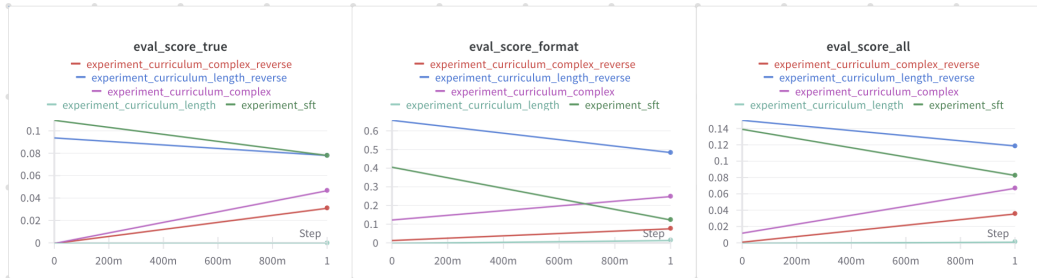


Figure 4: Visualization of countdown correctness score, format score and combined score over the course of training with the SFT curriculum learning extension for up to 2 epochs.

5 Discussion and Conclusion

Our results do not provide evidence for a benefit of curriculum learning over vanilla SFT. However, this is likely due to significant limitations of the SFT dataset used here. More specifically, training examples with longer completions and longer answer strings in the SFT dataset tended to be more likely to be *incorrect*. Qualitative evidence of this is shown in Figure 1. Thus, our curriculum learning approach was effectively to train on a set of good examples followed by bad examples. So it is unsurprisingly that we observed that curriculum learning impaired performance and that we actually saw a slight performance benefit of ‘reverse’ curriculum learning over vanilla SFT. If we had been provided with a better SFT dataset or if we had implemented some filtering steps to ensure that all of the examples in the SFT dataset are correct, we may have seen benefits of curriculum learning.

Our exploration of different sampling strategies here suggests that choosing the temperature and number of beams for beam search carefully during generation can substantially improve performance.

6 Team Contributions

- **Ethan Trepka** Implemented all the algorithms, generated figures and wrote up results.

Changes from Proposal Focused specifically on curriculum learning, and also added a second mini-extension focused on exploring different sampling strategies.

References

- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) (*ICML '09*). Association for Computing Machinery, New York, NY, USA, 41–48. <https://doi.org/10.1145/1553374.1553380>
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. arXiv:arXiv:2503.01307
- Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2021. Curriculum Learning: A Survey. arXiv:arXiv:2101.10382