Extended Abstract

Motivation This work is motivated by the observation that the order in which cognitive strategies are learned remains underexplored. Inspired by human learning stages, we aim to investigate whether constructing a curriculum, based on data difficulty or strategy sequencing, can improve performance during supervised fine-tuning (SFT) and direct preference optimization (DPO) training.

Method We experimented curriculum construction in both SFT and DPO stages for Countdown task. For DPO, data were pre-ranked by difficulty and fed in sequence. For SFT, we organized datasets by cognitive strategy type and tested the effect of different strategy orderings.

Implementation Implementation was done in PyTorch and trained on 32–48 GB GPUs using gradient accumulation to support larger batch sizes, either on a single AWS g6e instance or a Tesla V100-PCIE-32GB.

Results DPO showed unstable loss and yielded no meaningful improvement, the most likely issue being the inability to generate enough correct samples, which causes the drastic decrease in performance and the negative feedback. Incorporating strategy-annotated datasets in SFT degraded performance compared to baseline, and the model often stuck in one number and cannot proceed.

Discussion The experiments were mostly limited by computational resource. During DPO on-policy training, a key issue was the inability to break ties between similarly poor samples, which may be improved by adding components to the score function or adding external interventions to correct for the errors. Additionally, pre-computing a preference dataset with naive approach will not work either. Lastly, varying the order of cognitive strategies during SFT was ineffective; a more promising direction may involve dynamic strategy selection rather than static ordering.

Conclusion It is crucial to carefully curating and pre-evaluating dataset quality. Availability of ground truth preferences or rewards could significantly improve training outcomes.

Exploring Curriculum Learning in Different Stages of Large-Language Model Fine-tuning

Jiaqi Wu
Department of Bioengineering
Stanford University
jiaqiw01@stanford.edu

Abstract

This work explores whether sequencing cognitive strategies or ordering data by difficulty can improve model training in supervised fine-tuning (SFT) and direct preference optimization (DPO). Inspired by human learning stages, we investigate curriculum design through both pre-computed data difficulty and structured input with different orderings of cognitive strategy. Experiments were conducted using PyTorch on 32–48 GB GPUs with batch accumulation. In the DPO setting, ranking data by unequal scores is crucial, without which the performance deteriorates drastically. However, the amount of samplings needed for different scores led to days of training for a simple epoch. Similarly, varying the order of cognitive strategies during SFT failed to yield improvements and may worsen results. These outcomes suggest limitations in current reward formulations and highlight the importance of high-quality, pre-evaluated datasets. Future work may benefit from dynamic strategy selection or more discriminative reward signals.

1 Introduction

Curriculum learning in reinforcement learning (RL) is a training paradigm where agents are exposed to a structured sequence of tasks that increase gradually in complexity. This idea is inspired by the way humans and animals learn—by first mastering simple skills before progressing to more difficult ones. In the context of RL, where agents must learn through trial and error and often face sparse or delayed rewards, curriculum learning plays a crucial role in guiding exploration and accelerating policy optimization Soviany et al. (2022). By starting with easier subtasks and incrementally increasing difficulty, a curriculum can help stabilize training, improve sample efficiency, and prevent agents from getting stuck in poor local optima. This is especially important in real-world RL applications, where task complexity, environment noise, and limited feedback make direct learning from scratch highly inefficient or even infeasible. Furthermore, curriculum learning has been shown to reduce memorization of noisy or mislabeled samples, enhancing model robustness—particularly when training data is imperfect or collected from uncurated sources such as the web Castells et al. (2020). As RL continues to scale toward increasingly complex domains, curriculum learning offers a principled approach to structure learning and boost generalization.

Building on the importance of structured learning, another critical dimension of curriculum design in reinforcement learning is the diversity of training data. Integrating data diversity into the curriculum framework opens up opportunities to systematically expose models to varying types of inputs across different stages of training. This encourages us to incorporate augmentation strategies—such as inducing chain-of-thought reasoning, injecting AI-generated feedback, or artificially construct preference datasets. For instance, light augmentations introduced early on may help the model generalize syntactically, while more challenging transformations applied in later stages can promote robustness and expressive generation. By tailoring augmentation intensity to the progression of the

curriculum, we can also shape the agent's exposure to linguistic variability in a staged manner. This stage-aware interaction between augmentation and exploration not only enhances output diversity but can also improve generalization across tasks, domains, and user intents, ultimately influencing the effectiveness and adaptability of the learned policy.

Here, I focus on Countdown task and aim to integrate diverse types of fine-tuning datasets—each representing a distinct curriculum—into a unified training pipeline for LLMs. These datasets, varying in complexity and difficulty, will be introduced at different stages of training (e.g., across epochs or steps), enabling dynamic modulation of reward weighting, as well as exploration–exploitation balance. By experimenting with different curriculum structures and temporal schedules, the goal is to encourage the model to implicitly internalize the progression of task complexity, thereby enhancing its sampling diversity, reasoning sophistication, and generalization capabilities.

2 Related Work

Curriculum learning has been explored through various mechanisms, including reinforcement learning-based approaches. For instance, Kumar et al. (2019) propose a meta-learning curriculum strategy for neural machine translation, where a noise estimator is trained to assess the difficulty of training examples. Based on these difficulty scores, the dataset is partitioned into multiple bins. The authors then employ a Q-learning-based reinforcement learning agent to dynamically select which bin to sample from during training. Crucially, their method incorporates an exploration mechanism that ensures all bins are initially visited—promoting coverage and diversity—before gradually shifting toward exploitation of the learned policy. This strategy mirrors an annealing process in which the curriculum becomes more focused and specialized over time. However, this work did not focus on LLM finetuning, and the policy function they used may be too simple. Similarly, Kim and Lee (2024) applied curriculum learning to LLMss by ordering training data from simple to complex based on prompt length, attention scores, and loss values. The results showed that curriculum learning slightly improved LLM performance compared to random data shuffling, with attention-based ordering yielding the best gains. This approach enhances scalability and efficiency in LLM training without increasing model or dataset size

Building on cognitive motivations, Gandhi et al. (2025) explore curriculum design through the lens of human-like reasoning behaviors. They inject cognitive structures, such as verification, backtracking, subgoal decomposition, and backward chaining, into LLMs by priming them with exemplars that exhibit these patterns. Using synthetic datasets designed to highlight such behaviors, they demonstrate that even minimal fine-tuning with cognitively-structured reasoning chains can substantially enhance a model's problem-solving ability. This work suggests new directions for curriculum and data augmentation strategies that scaffold learning through increasingly sophisticated forms of human-like cognition. However, the paper does not directly compare or prescribe a specific order in which the four cognitive strategies should be learned. Instead, it systematically investigates the impact of each strategy and their combinations by priming models with datasets that emphasize different behavioral subsets before reinforcement learning. This provides a possible direction in connecting the strategies into a curriculum, where learning one before the other may be beneficial than the others.

Curriculum is also embedded in other aspects such as training losses. SuperLoss Castells et al. (2020) introduces curriculum learning into model training by dynamically adjusting the influence of each training sample based on its current loss value. The approach is task-agnostic and does not require manual design or prior knowledge about the difficulty of samples. Instead, SuperLoss designs the architecture such that samples with higher loss (i.e., harder or potentially noisy samples) are automatically downweighted, meaning their impact on the model update is reduced. Conversely, easier samples (those with lower loss) contribute more strongly to the model's learning early on.

2D-Curri-DPO introduces a two-dimensional curriculum learning framework for Direct Preference Optimization (DPO) Li and Zhang (2025). This framework pre-computes each training sample along two axes: prompt complexity (PC) and pairwise distinguishability (PD). The former measures how semantically complex or challenging a prompt is, estimated by generating multiple candidate responses to a prompt using a reference model, then evaluating the perplexity of each response under a fixed language model. The latter compares the quality score by an external judge or scoring function from sample pairs.

Similar to the ideas of separating data into multiple-bins and incorporate implicitly different learning stages/curriculum, DeepSeek-R1 DeepSeek-AI et al. (2025) incorporates multi-stage training and cold-start data before RL, which is closely related with following a curriculum through different stages of learning. After starting with cold start data, phase enhancing model's reasoning capabilities using reasoning-incentive tasks such as coding, mathematics. After reasoning-oriented RL converges, SFT stage where data from other domains such as writing, role-playing are incorporated. They adopted a Group Relative Policy Optimization. To model the reward, accuracy reward and format rewards are used, which enforces the model to put its thinking process between 2 think tags. Although this is a sophisticated work with lots of experimentation, having different training stages, checkpoints can be time and memory consuming. It may be beneficial for us to embed and unify different training stages into 1 training/finetuning, which is the focus of the proposed work.

3 Method

Due to the high computational cost observed during initial exploration, we investigated two strategies to mitigate resource demands in the DPO stage. The first strategy involved pre-computing all preference labels prior to training, thereby avoiding the need for VLLM-based sampling during optimization. In parallel, we also experimented with different dataset arrangements and prompt structures in the SFT stage. For SFT, we used the dataset introduced by Gandhi et al. (2025), while the dataset used for DPO and RLOO experiments was derived from TinyZero Pan et al. (2025).

3.1 DPO on-policy sampling with data of progressive difficulty

This approach follows what is given on the project guideline. The additional consideration is the arrangement of dataset using a naive pre-computed difficulty level as below:

$$\alpha \times \textbf{num\_operators} + \beta \times \textbf{distance} \tag{1}$$

where *distance* refers to the absolute difference of the best result from the simple depth-first search from the target.

Due to computational time, only 3000 samples from each difficulty quantile were selected. But this takes >10 hours per epoch. The distribution of dataset difficulty is shown in Figure 1 below.

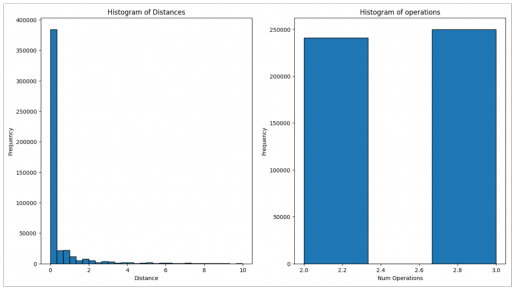

Figure 1: Distribution of the difficulty level for TinyZero dataset

3.2 Pre-compute Preference Samples for DPO

Considering the training inefficiency for on-policy sample generation, we considered pre-computing a preference dataset using both the 1k SFT training dataset and also the TinyZero dataset. For SFT dataset, since the solution is given, the artificial rejected samples are obtained by arbitrarily creating off-by-one error in the numbers used during the process or operators in the solution. For TinyZero dataset, samples with different scores are sampled from the base model (model after SFT stage) beforehand, and that samples with higher score is mareked as preferred.

3.3 SFT with Separate Cognitive Strategies

We explored how different cognitive strategies influence model learning by fine-tuning with prompts annotated according to three reasoning styles introduced in Gandhi et al. (2025):

1. Backtracking-only, where the model explores solution paths and retreats when encountering dead ends

2. Backtracking-backward, where the model works backward from the goal state to the initial state

3. Backtracking-subgoal, where the model breaks down complex problems into manageable intermediate steps.

Each strategy type exhibits distinct reasoning patterns. For instance, in backtracking-backward, the model attempts combinations with clear reference to the goal, such as:

Let me try different combinations:
First, let me try with 19 and 14:
19 - 14 = 5 (getting closer to 13)
I have 5, 3, and 2 now:
5 + 3 = 8 (still need 5 more)

This reflects a trial-and-error process guided by the goal value.

In backtracking-subgoal, the model reasons via intermediate objectives: *Let me try to reach some subgoals:*
Let's try to reach 26 since it's a multiple of 13.
19 + 14 = 33
Let's try a different approach..

3.4 Evaluation Metric

The score function from Gandhi et al. (2025), integrating a 0.1 of format score and 1.0 for correct format and result were used as evaluation metric. The test dataset is form the 200 test cases in cog-all-strategy dataset. Sampling were done with temperature = 0.6, topK = 20, and topP = 0.95.

4 Experimental Setup

For DPO, a single AWS g6e instance with approximately 48GB of GPU memory was used. The maximum batch size was 1, or 2 when using LoRA. Gradient accumulation steps ranged from 16 to 32. Input prompts were limited to 256 tokens, and responses to 1024 tokens. VLLM sampling was terminated upon encountering the </answer> tag. The dataset was organized by difficulty level, as described in the previous section.

After observing no significant improvement with DPO, we shifted focus to supervised fine-tuning (SFT). The original prompt–answer pairs, which explicitly encode distinct cognitive strategies, were obtained using the official GitHub scripts from Gandhi et al. (2025).

To investigate the impact of strategy sequencing, we constructed a combined dataset containing 1,000 training samples per strategy, ordered as backtracking-only, backtracking-backward, and backtracking-subgoal (totaling 3,000 samples). The model was trained for epoch // 3 epochs on each phase to ensure equal exposure to each strategy. This was followed by an additional 2 epochs of training on a mixed dataset containing all three strategies (cog-all-strategy), resulting in a total of 10,000 training samples across all training stages (based on epoch × dataset size).

As a baseline, an additional model was trained on the same data with prompts randomly assigned to different cognitive strategies, without any fixed ordering or structure.

Table 1: SFT training configurations with different cognitive strategy sequences. 123 represents backtracking-only -> backtracking-backward -> backtracking-subgoal

SFT Dataset Sequence	Epoch	Total Training Samples (epoch × sample size)
123 - all-strategy	2–2	10k
213 - all-strategy	2–2	10k
312 - all-strategy	2–2	10k
Randomized - all-strategy	2–2	10k
All-strategy only	10	10k

5 Results

5.1 DPO Implementation

We observed minimal learning during the DPO stage. The model frequently reused the same number twice in the final answer—a systematic error that persisted across samples and quickly degraded performance. When most generated samples received low scores (e.g., 0 or 0.1), the overall solution quality collapsed toward 0.1. Although generating a correct sample could improve the batch quality, doing so often required over 500 sampling attempts per data point.

Additionally, pre-computing a preference dataset for DPO by introducing arbitrary errors into the SFT dataset failed to yield any performance improvement.

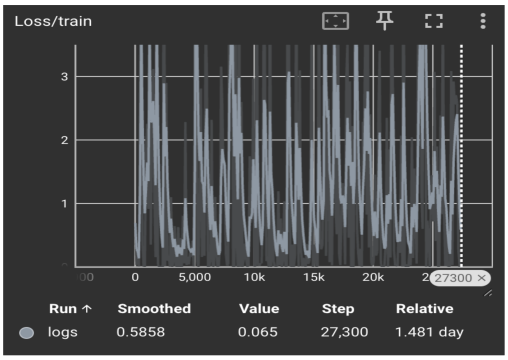

Figure 2: Example of DPO loss without sampling a correct sample

5.2 SFT Extensions

5.3 Quantitative Evaluation

Table 2: Results for SFT with different cognitive strategy sequences over 200 test samples. 123 represents backtracking-only -> backtracking-backward -> backtracking-subgoal

SFT Dataset Sequence	Epoch	Average Test Score
123 - all-strategy	2–2	0.10
213 - all-strategy	2–2	0.12
312 - all-strategy	2–2	0.11
Randomized - all-strategy	2–2	0.10
All-strategy only	10	0.24

5.4 Qualitative Analysis

With the inclusion of solutions using distinct cognitive strategies, some of the erroneous response is shown below:

Let me work to reach 56 from 53.
53 is 56 from 53.
53 is 53.
11 is 53.
59 is 11.
86 is 11.
59 is 11.
11 is 53.

Let me try to reach some useful subgoals:
- 2 31 is 51, which is 51
- 49 is 51, which is 51
- 58 is 21, which is 21
- 21 is 51, which is 51
- 51 is 51, which is 51
- 21 is 51, which is 51
- 51 is 51, which is 51
- 51 is 51, which is 51

We can see that if the model starts the answer with subgoal strategy, it often fail by stucking in 1 number and looping forever. The same issue is observed when it chooses the backtracking-backward strategy. It seems that once the starting point is incorrect, the error becomes unamendable.

Let me work backwards from 61.
61 could be reached by 71.
71 could be reached by 61.
61 could be reached by 61.
61.
61.
61.

6 Discussion

As seen previously, mingling different cognitive strategies can hurt performance, confuse the model and lead to poor performance. Instead, a more systematic and top-down approach should be employed. Also note that Qwen may have cognitive ability from the beginning Gandhi et al. (2025), which makes it less suitable for studying the sequence of strategies and using specific strategies for fine-tuning. Thus, a future direction is to use other models such as llama. Additionally, we note that even if different sequencing of strategies lead to difference in performance, it will be difficult to conclude whether the improvement is caused by the more diverse answer text or the model actually emplying different strategies.

Apart from the limitations imposed by computational resources, several practical challenges were encountered during DPO training. First, the model often produced identical scores for different responses, which reduced gradient signal and slowed learning. When filtering out samples with duplicate scores, it required up to 400 sampling attempts (and sometimes infinite looping) per instance to obtain satisfactory training pairs, significantly increasing computation time. Moreover, the approach lacked a structured mechanism to assess response quality, such as comparing the numerical distance to the target answer or penalizing the severity of mathematical errors (e.g., invalid equations). A more effective strategy would involve identifying common systematic errors—such as the repeated use of a number in the solution—and dynamically modifying the prompt to discourage such behavior. Additionally, relying on reinforcement from "easy" samples until the model achieves satisfactory performance can be inefficient, as it may take hundreds or even thousands of on-policy samplings without explicit intervention or curriculum design.

Through the course of experimentation, we uncovered several important insights and limitations that highlight the complexity of applying DPO in this setting. First, pre-computing the preference dataset

using a simplistic method led to performance deterioration. This likely stems from the rejected samples being too easily distinguishable from the chosen ones, reducing the informativeness of preference signals and revealing the model's sensitivity to dataset quality. Furthermore, as shown in Figure 1, the distribution of difficulty levels is highly skewed, with the majority of samples categorized as "easy," which may bias the learning process and limit generalization.

Another key challenge lies in the model's ability to determine which cognitive strategy to apply when faced with a novel math task. This decision process remains opaque, suggesting that a more structured or top-down control mechanism may be beneficial. For example, Akella (2024) proposed a lightweight classifier to first categorize problems, followed by strategy selection conditioned on the predicted category. Inspired by this, a promising direction for future work could involve integrating a reward signal for appropriate strategy selection—even when the final answer is incorrect—thereby encouraging more meaningful preference modeling. However, we may also consider that if we were to mimicking human problem solving, the model should be able to try every method they have and there does not need to have a hard classification of problem type.

There are many interesting future research regarding error correction and design of reward function with more than just format and correctness scores. For instance, setting up the curriculum learning via a multi-agent architecture for error correction (like a teacher-student setting) and change of problem solving strategy may be promising.

7 Conclusion

Although experimenting with curriculum in the DPO stage has not been successful, it calls attention to the need for more principled approaches in preference data construction, strategy selection, and reward design. Our findings highlight the model's sensitivity to the quality and difficulty balance of the preference dataset, as overly simplistic rejected samples diminish learning signals. Additionally, the skewed distribution of sample difficulty underscores the importance of dataset curation in preference-based training.

The model's inability to explicitly select an appropriate cognitive strategy also points to the potential of a top-down framework—such as using a lightweight classifier to predict the best strategy or incorporating a reward function that encourages correct strategy use, even when the final answer is incorrect. These observations emphasize that many of the core challenges and design insights only emerge through iterative experimentation, underscoring the value of hands-on development in refining preference-based alignment methods.

8 Team Contributions

Only one person in the group.

Changes from Proposal I initially aimed to explore incorporating exploration–exploitation trade-offs directly into the reward or loss function. For example, a complexity or cognitive-load score could be assigned to each sample and used to scale its reward. Within the Bradley-Terry framework, such auxiliary features could inform pairwise comparisons by weighting losses to favor correct reasoning under higher difficulty or novelty. However, due to the baseline DPO setup failing to yield satisfactory results and the excessive time required for sampling, I was unable to pursue these more sophisticated approaches. Instead, I focused on heuristic-based difficulty estimation and dataset separation during both the DPO and SFT stages. I also chose not to explore RLOO, as the slow sampling in DPO suggested that RLOO training would also be inefficient.

References

Amogh Akella. 2024. Improving Math Problem Solving in Large Language Models Through Categorization and Strategy Tailoring. arXiv:2411.00042 [cs.CL] `https://arxiv.org/abs/2411.00042`

Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. SuperLoss: A Generic Loss for Robust Curriculum Learning. In *Advances in Neural Information Processing Systems*,

H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 4308–4319. https://proceedings.neurips.cc/paper_files/paper/2020/file/2cfa8f9e50e0f510ede9d12338a5f564-Paper.pdf

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

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:2503.01307 [cs.CL] https://arxiv.org/abs/2503.01307

Jisu Kim and Juhwan Lee. 2024. Strategic Data Ordering: Enhancing Large Language Model Performance through Curriculum Learning. arXiv:2405.07490 [cs.CL] https://arxiv.org/abs/2405.07490

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. 2019. Reinforcement Learning based Curriculum Optimization for Neural Machine Translation. arXiv:1903.00041 [cs.CL] https://arxiv.org/abs/1903.00041

Mengyang Li and Zhong Zhang. 2025. 2D-Curri-DPO: Two-Dimensional Curriculum Learning for Direct Preference Optimization. arXiv:2504.07856 [cs.AI] https://arxiv.org/abs/2504.07856

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum Learning: A Survey. arXiv:2101.10382 [cs.LG] https://arxiv.org/abs/2101.10382

A Additional Experiments

RLOO was implemented and can be run successfully. However, I have not had enough time to train it and get test results. After 8 hours of training the loss curve with k = 5 and batch = 1 accumulated over 32 steps:

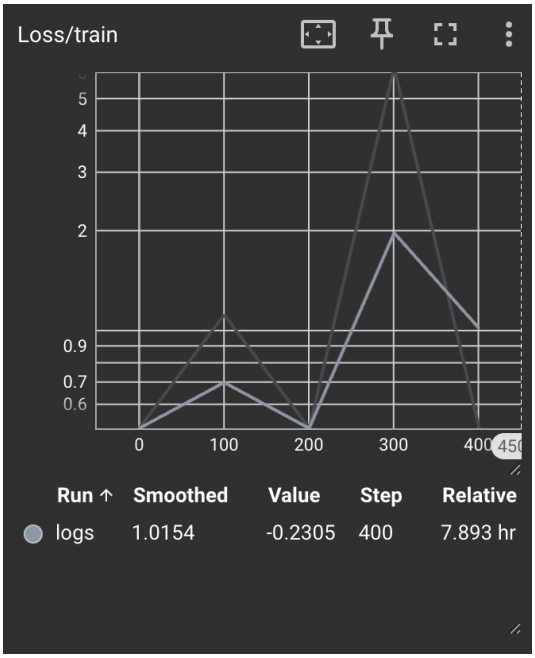

Figure 3: RLOO training loss for 400 steps