BREAD: Branched Rollouts from Expert Anchors Bridge SFT & RL for Reasoning

Xuechen Zhang [*1] **Zijian Huang** [*1] **Yingcong Li** [1] **Chenshun Ni** [1] **Jiasi Chen** [1] **Samet Oymak** [1]

Abstract

Small language models (SLMs) struggle to learn complex reasoning behaviors, especially when high-quality traces are scarce or difficult to learn from. A typical approach for training such models combines a supervised fine-tuning (SFT) stage, often to distill reasoning capabilities from a larger model, followed by a reinforcement learning (RL) stage such as Group Relative Policy Optimization (GRPO). In this paper, we investigate the fundamental limitations of this SFT + RL paradigm and propose methods to overcome them. Using a toy student-expert model over Markov chains, we demonstrate that the SFT + RL strategy can fail completely when (1) the expert's traces are too difficult for the small model to express, or (2) the small model's initialization achieves exponentially sparse rewards as task complexity grows. To address these, we introduce BREAD, a GRPO variant that bridges SFT and RL via partial expert guidance and branch rollouts. When self-generated traces fail, BREAD adaptively inserts short expert prefixes/hints, allowing the small model to complete the rest of the reasoning path, and ensuring that each update includes at least one successful trace. This mechanism both densifies the reward signal and induces a natural learning curriculum. BREAD requires fewer than 40% of ground-truth traces, consistently outperforming standard GRPO while speeding up the training by about 3×. Importantly, we find that BREAD helps the model solve problems that are otherwise unsolvable by the SFT + RL strategy, highlighting how branch rollouts and expert guidance can aid SLM reasoning.

[*]Equal contribution [1]EECS department, University of Michigan, Ann Arbor, USA. Correspondence to: Firstname1 Lastname1 <first1.last1@xxx.edu>, Firstname2 Lastname2 <first2.last2@www.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Over the past few years, we have witnessed a significant push toward enhancing language model reasoning, which has led to highly capable frontier models such as OpenAI o1 (Jaech et al., 2024), Gemini 2.5 (Kavukcuoglu, 2025), and DeepSeek R1 (Guo et al., 2025). These models can generate longer chain-of-thought (CoT) traces and utilize more test-time compute to tackle challenging tasks (Muennighoff et al., 2025). Despite these innovations, reasoning with small language models (SLM) remains a challenge. For instance, DeepSeek-R1 (Guo et al., 2025) has 671B parameters whereas the distilled model sizes range from 1.5B to 70B, and their performance substantially degrades at the 1.5B model (see Table 5 in (Guo et al., 2025)).

This work studies optimization strategies to enhance SLMs, with emphasis on reasoning tasks. Two popular optimization strategies for training LLMs are supervised fine-tuning (SFT) and reinforcement learning (such as GRPO or proximal policy optimization). Often, an SFT phase is employed, followed by an RL phase. While this two-stage procedure has found success, for SLMs, long context reasoning problems can pose unique challenges due to the misalignment between the expert and student models and potentially sparse rewards. For example, consider a scenario in which each token generated by an expert/teacher model requires the smaller student model to produce K intermediate tokens to express it. In other words, the expert thinks and outputs K steps ahead, from the student's point of view. In practice, this situation can arise when the expert model is a $\times K$ deeper version of the student. Naturally, such expert traces might be too challenging for the small model to learn from[1]. On the other hand, success of the RL phase often relies on a good initialization during the SFT phase. In Appendix B.1, we provide a mathematical setting capturing this intuition and demonstrate that SFT+RL can fail for small models (see Figure 5), especially on difficult problems.

To address the difficulties of small models in learning from complex traces during fine-tuning, we propose our algorithm, Branch Rollouts and Expert Anchors for Densified

[1]In practice, SFT+RL can work well for much of the dataset but might fail on a subset of a difficult problems. See Section 3.1 for empirical evidence and evaluations.

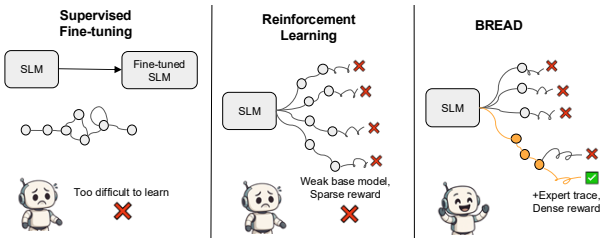


Figure 1. High-level overview of approaches. In existing training methods like supervised fine-tuning (left), high-quality reasoning traces produced by LLMs are often too complex for SLMs to imitate, so they deliver little benefit and can even hurt SLM reasoning capability. Since the subsequent RL phase starts from this weak starting point, the two-stage SFT+RL procedure often fails. In reinforcement learning (center), when the initial policy generates incorrect traces, the rewards are sparse, causing slow or ineffective learning. We propose BREAD (right), which uses part of an expert trace as an anchor, then generates additional rollouts that branch from its intermediate episodes. These branched trajectories provide denser, higher-quality feedback, helping SLMs learn robust reasoning strategies. Each dot represents a single episode, and the yellow trajectory is the expert trace.

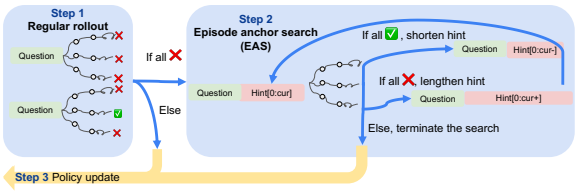


Figure 2. Workflow of BREAD. (1) **Regular rollout**: Given a question Q, sample a group of rollouts. If the sampled rollouts contain correct reasoning trace, use this group of rollouts to do the **policy updates**. Otherwise, go to step (2) **Episode anchor search**: Starting with the whole expert trace (provided by the ground truth or from the correct responses generated by LLMs) as the search space for potential suitable hints, construct a hint using the first half of the expert trace, append it to the question q, and sample a group of rollouts. If all the rollouts are correct, shorten the hint to contain fewer episodes and repeat the process; if all rollouts are wrong, lengthen the hint; otherwise, use the current rollouts to do the **policy update**.

rewards, depicted in Figure 1. BREAD gracefully integrates the SFT and RL phases by anchoring the optimization process with the expert traces, while allowing the small base model to acquire progressively more flexibility as it becomes a stronger problem solver. Assuming an expert trace is available (e.g. by querying a large expert model), BREAD updates the model with a correct trace; however, its traces are progressively more self-generated.

Contributions: Our specific contributions are as follows:

- **Methodology:** We introduce BREAD as a GRPO variant integrating SFT and RL phases, while inducing a learning curriculum along the reasoning trace. To motivate BREAD, we introduce a toy student-expert model using Markov chains. This task demonstrates how the expert's trace can be uninformative for the student model and how the subsequent RL phase can fail due to sparse rewards. In contrast, BREAD solves this task efficiently.
- **Empirical impact:** Our experimental results show that BREAD matches or surpasses SFT + vanilla GRPO while using $\approx 20\%$ of the correct trace tokens. By reducing the number of rollouts and the total optimization steps, BREAD lowers overall training compute by $\approx 75\%$ relative to vanilla GRPO. Additionally, we construct a slice of difficult problems and demonstrate that BREAD, when trained over this set, achieves substantially higher accuracy compared to SFT+GRPO. Finally, we provide an empirical study of how branching rollouts from expert hints densify the reward signal.

The rest of the paper is organized as follows: Appendix A discusses the related work on language model reasoning and traditional RL methods. Section 2 explains our algorithm BREAD, which also contains the observations inspiring our algorithm design. Section 3 presents and discusses our main experiment results to demonstrate the effectiveness of BREAD. Appendix G concludes the paper and discusses the limitation and future directions to improve in this domain.

2. Proposed Method: BREAD

In this work, we propose the **B**ranch **R**ollouts and **E**xpert **A**nchors for **D**ensified RL (BREAD) algorithm. We will first describe the algorithm at a high level, followed by a mathematical toy model example (Appendix B.1), and the key observations (Appendix B.2, Appendix B.3) that support its design.

In BREAD, for each question q paired with the answer a, the workflow of BREAD is shown in Figure 2 and proceeds as follows:

1. **Regular rollout:** Sample a group of G rollouts $\{o_i\}_{i=1}^{G}$.
2. **Episode anchor search:** If the success rate of the initial group is too low (e.g. lower than a threshold), do a binary search to find a short hint ρ that contains the "Expert Anchor" in the expert solution. "Expert Anchor" means the success rate of a new sampled output group $\{o_i'\}_{i=1}^{G}$, resulting from the question appended with the hint from the expert trace (q, ρ), is within a pre-defined range.
3. **Policy updates:** Optimize the policy via the following objective:

$$\mathcal{J}_{\text{BREAD}}(\theta) = \mathbb{E}_{(q,\rho,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|(q,\rho))} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}(\min(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_{i,t}) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})) \right], \tag{1}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,\rho,o_{i<t})}{\pi_{\text{old}}(o_{i,t}|q,\rho,o_{i<t})}, \quad \hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)} \tag{2}$$

The details of the algorithm can be found in Algorithm 1. Next, we will discuss the key observations that motivate the design of BREAD.

3. Experiments

Baseline algorithms. We evaluate the effectiveness of our method BREAD on mathematical tasks, which can be easily adapted to other reasoning tasks such as coding, commonsense reasoning, etc. We compare with the following baselines:

- `GRPO`: Standard GRPO.
- `SFT`: Standard supervised fine-tuning on the full training set. We denote SFT on various data splits as `SFT(X)`, e.g. `SFT(full)` for the whole dataset, `SFT(difficult)` for the hardest subset, `SFT(random)` for a size-matched random subset, and `SFT(selected)` for the EAS-chosen subset. The precise discussion is detailed with the experiments results.
- `SFT + GRPO`: This means `SFT` is run followed by `GRPO`. In other words, `GRPO` continues from the final checkpoint of the same `SFT` run. We denote GRPO fine-tuning initialized from an SFT model trained on data split X as `SFT(X) + GRPO`.
- `GRPO w/ Expert Trace`: During `GRPO` training, the entire expert trace is injected as an additional rollout for all queries. This is a strong baseline as it contains the expert. Details in Appendix E.

Training settings. We adopt the verl framework (Sheng et al., 2024) for training. We utilize the Adam optimizer (Kingma, 2014) with a constant learning rate of 1×10^{-6}. For rollout, the prompt batch size is 256 and we sample 8 responses for each prompt. For training, the mini-batch size is 64. To find the appropriate branching point during the Episode Anchor Search (EAS) step of BREAD (Section 2), we first split the expert trace into sentences (on easier datasets like MATH (Hendrycks et al., 2021), where the expert traces are generally short) or into paragraphs (on harder datasets like NuminaMth-CoT (LI et al., 2024) and OpenR1-Math-220K (Face, 2025)). Then, we aggregate the split partitions into $K = 10$ episodes evenly, and proceed with binary search. See details in Appendix D.3.

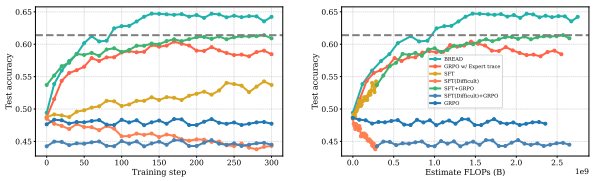
3.1. Main Results

BREAD outperforms all baselines in terms of test accuracy. Figure 3 plots the training curves of all methods on the NuminaMath-CoT benchmark, starting from the Qwen-2.5-3B-Instruct base model. In Figure 3a, we visualize the test accuracy against training steps for BREAD and the baseline methods. BREAD outperforms all the baselines. Let us discuss each baseline in turn. BREAD improves final accuracy by more than 15% over vanilla `GRPO`. SFT can offer a stronger starting point for RL, which we can see by comparing the `SFT` and `SFT+GRPO` curves, but BREAD is still better. (Note that training SFT for more iterations does not further improve performance, see Appendix D.1.) `SFT(Difficult)` represents SFT trained on the hardest questions from NuminaMath-CoT, obtained by sorting samples by solution length. Intuitively, problems that require longer solutions are typically more complex and involve more reasoning steps. Comparing the `SFT(Difficult)` and `SFT(Difficult)+GRPO` curves, we can see that SFT with too complex expert traces can even hurt the performance of SLM, which further hurts the later GRPO stage. Finally, from the `GRPO w/ Expert trace` curve, we see that adding the expert trace as an extra rollout during GRPO mitigates sparse rewards and shows noticeable performance improvement, but it is still worse than BREAD. We suspect this is because there is a distribution gap between small and large model's reasoning traces, which makes SLM imitation of large models hard, while BREAD can reduce the gap during learning by letting the SLM figure out the reasoning steps by itself more. See further discussion below on **"hard questions"** for elaboration. We also provide more results among different base models and datasets in Table 1 and Appendix D. **BREAD reduces training time.** BREAD is also markedly more training-efficient. It can reach the accuracy of the best baseline, `SFT+GRPO`, in just 25% of the training steps (Figure 3a), translating to roughly a 75% reduction in total compute FLOPs (Figure 3b). We estimated the FLOPs based on the common cost evaluation method used in recent scaling-law studies (Snell et al., 2024; Hoffmann et al., 2022; Sardana et al., 2023), by counting a forward pass as $2ND$ and a backward pass as $4ND$. Here N is the number of model parameters and D is the total token count processed in that pass. For calculation details, see Appendix C.1. Note that we estimate D as the average length of all expert traces in the training set, but according our measurements, BREAD's actual generation length is always shorter. Therefore for BREAD, its actual token count, and thus its FLOP cost, is even lower than the estimated values reported Figure 3b. We also discuss the potential of reducing the number of rollout needed to reduce training cost in Appendix D.2.

BREAD succeeds on the hard questions, where other baselines fail. To understand the gains of BREAD, we train and evaluate each method exclusively on very difficult problems. The goal is to investigate whether expert hints in BREAD can help SLMs learn new information particularly from these hard questions. We conduct the experiment on the NuminaMath-CoT dataset and the Qwen2.5-3B-Instruct as the base model. To build the hard dataset, we first run ordinary SFT and from the training dataset, we select the 500 questions for which three independent generations produce no correct trace (pass@3 = 0). These tasks are unsolved by the small model, and their expert traces proved too complex

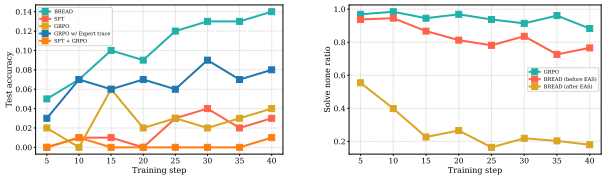
Model	Dataset	GRPO	SFT(full)+GRPO	SFT(random)+GRPO	SFT(selected)+GRPO	BREAD	DeepSeek-Distill
Qwen-1.5B-Instruct	MATH	0.590	0.774	0.712	0.706	0.788	0.818
Qwen-3B-Instruct	MATH	0.681	0.846	0.735	0.794	0.843	/
Qwen-1.5B-Instruct	NuminaMath-CoT	0.347	0.242	0.356	0.234	0.361	0.368
Qwen-3B-Instruct	NuminaMath-CoT	0.475	0.537	0.519	0.502	0.647	/

Table 1. BREAD outperforms other baselines and nearly reaches the accuracy of DeepSeek-R1-Distill, which has the benefit of vast training data. There is no DeepSeek-R1-Distill for Qwen-3B provided by (Guo et al., 2025), so its cells are left blank.



(a) Test acc over training steps (b) Test acc vs estimated FLOPs

Figure 3. Test accuracy over training steps (left) / FLOPs (right). BREAD, which adaptively uses hints from expert traces during GRPO, significantly improves SLM reasoning ability compared to all baselines. The gray dashed line (max accuracy of the best baseline) demonstrates that BREAD can speed up the convergence speed by about 3×. Both figures share the same legend.



(a) Test acc over training steps (b) Solve none ratio

Figure 4. (a) Test accuracy on very hard questions over training steps. BREAD outperforms other baselines significantly while the traditional methods (`SFT`, `GRPO`, `SFT+GRPO`) learn little. (b) Proportion of test questions for which every rollout fails (solve-none ratio). In vanilla GRPO, the ratio stays persistently high, signalling that training stalls under sparse rewards. In BREAD, the solve-none ratio for regular rollout starts similarly high, but EAS injects expert traces and densifies the reward. This rate continuously decreases, confirming that the model is learning.

for SFT to learn. The 500 samples were split into 80/20 train/test subsets. We then again used Qwen2.5-3B-Instruct as the base model and trained each method on this hard subset. The results are shown in Figure 4a. All baselines show little or no improvement on the test set after training on the hard questions. In contrast, BREAD achieves a clear upward performance with continued training, demonstrating that its partial expert guidance and branched rollout strategy can provide learnable information, even when standard `SFT` and vanilla `GRPO` fail. We argue that BREAD succeeds because it adaptively reduce problem difficulty and densifies the rewards. As shown in Figure 4b, BREAD sharply lowers the solve-none ratio, getting more informative samples and richer feedback. This enables BREAD to learn effectively even from very hard questions and complex reasoning traces.

BREAD improves sample efficiency during training. SLMs distilled via large-scale SFT can achieve strong reasoning capability, such as DeepSeek-R1-Distill (Guo et al., 2025). However, the distillation pipeline is prohibitively expensive. For example, this model is trained with 800k samples from both reasoning and non-reasoning domains curated with DeepSeek-R1, containing 671 billion parameters. A single forward pass through such a huge model already costs more FLOPs than 25 RL training steps with 8 rollouts. The pipeline also needs expensive sample filtering and trace post-processing, like the heavyweight data-collection procedure in (Muennighoff et al., 2025; Li et al., 2025). What makes things worse is that, as illustrated in Table 2, each target model requires training with different samples, multiplying the cost.

In contrast, BREAD is far more sample-efficient, achieving comparable gains with a small fraction of the expert trace and without any heavyweight sample selection stage. To show this, we created two trace-budget–matched baselines, `SFT(selected)` and `SFT(random)`. For fair comparison, we first record the expert traces actually requested by BREAD via Episode Anchor Search (`EAS`). On NuminaMath-CoT, this corresponded to 36.7% of samples, and on the easier Math (Hendrycks et al., 2021) dataset the fraction falls to 19.1% (during 300 training steps of Qwen-2.5-3B-Instruct). We then supervised fine-tune base models with the same number of traces To create `SFT(selected)`, we select the exact subset chosen by BREAD. To create `SFT(random)`, we use an equally sized randomly picked subset. Finally, we created `SFT(full)`, which uses all the expert traces, take a high cost. Each of the `SFT(x)` phases was followed by a GRPO phase. As the results in Table 1 show, BREAD outperforms nearly all the `SFT(X)+GRPO` baselines in terms of accuracy, and can even approach the performance of the expensive DeepSeek-R1-Distill model. Notably, while DeepSeek-R1-Distill gains from vast and diverse training data, BREAD achieves nearly comparable accuracy without requiring this data. We also observe that small models do not necessarily benefit from SFT with expert traces, as evidenced by the Qwen-2.5-1.5B-Instruct run on NuminaMath-CoT, where `SFT(full)+GRPO` has relatively low accuracy. Also, which samples are most useful for SFT is uncertain: on the MATH dataset, `SFT(selected)+GRPO` has higher accuracy than `SFT(random)+GRPO` with Qwen-3B-Instruct, but it is the opposite for Qwen-1.5B-Instruct.

Acknowledgements

This work is supported by the National Science Foundation grants CCF-2046816, CCF-2403075, CCF-2212426, the Office of Naval Research grant N000142412289, and an Adobe Data Science Research Award. The computational aspects of the research is generously supported by computational resources provided by the Amazon Research Award on Foundation Model Development.

References

Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R. J., Javaheripi, M., Kauffmann, P., et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Aggarwal, P. and Welleck, S. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590(7847): 580–586, 2021.

Face, H. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Google. Gemini 2.0 flash thinking mode (gemini-2.0f lash-thinking-exp-1219), 2024. https://cloud.google.com/vertex-ai/generative-ai/docs/thinking.

Google. Gemini 2.0 flash thinking mode (gemini-2.0-flash-thinking-exp-1219), December 2024. URL https://cloud.google.com/vertex-ai/generative-ai/docs/thinking-mode. Accessed 15 May 2025. "Last updated 14 May 2025" on the page.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Kavukcuoglu, K. Gemini 2.5: Our most intelligent ai model. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/, March 2025. Accessed 15 May 2025.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LI, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S. C., Rasul, K., Yu, L., Jiang, A., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., and Polu, S. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Li, Y., Yue, X., Xu, Z., Jiang, F., Niu, L., Lin, B. Y., Ramasubramanian, B., and Poovendran, R. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*, 2025.

Lin, Z., Lin, M., Xie, Y., and Ji, R. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Qu, Y., Yang, M. Y., Setlur, A., Tunstall, L., Beeching, E. E., Salakhutdinov, R., and Kumar, A. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Sardana, N., Portes, J., Doubov, S., and Frankle, J. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Wang, Z., Cui, G., Wan, K., and Zhao, W. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.

Xu, Y. E., Savani, Y., Fang, F., and Kolter, Z. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhang, X., Huang, Z., Ni, C., Xiong, Z., Chen, J., and Oymak, S. Making small language models efficient reasoners: Intervention, supervision, reinforcement. *arXiv preprint arXiv:2505.07961*, 2025.

A. Related Work

We provide further discussion of ”’ in Appendix ”’ Supervised fine-tuning (SFT) and Reinforcement Learning (RL). Recently, there is a debate about whether SFT or RL can truly improve the reasoning ability of language models (LMs). With the emergence of (Shao et al., 2024; Guo et al., 2025; Team et al., 2025), more and more reinforcement finetuned reasoning models demonstrate the importance of RL. in reasoning tasks. (Chu et al., 2025) prove that RL can enhance the reasoning ability of LMs while SFT can only force LMs to memorize knowledge. However, (Yue et al., 2025) argues that the base model already has the reasoning ability while Reinforcement Learning with Verifiable Rewards (RLVR) barely increases the probability of correct reasoning trace. Furthermore, distillation works such as (Guo et al., 2025; Muennighoff et al., 2025) prove that SFT can help SLMs acquire reasoning capability comparable to the expert/teacher models. While the current popular pipeline to train a reasoning LM is SFT followed by RL, we argue the need for stronger integration of SFT and RL because, for hard questions, the expert solution might not be suitable for base models to learn while RL can struggle to discover even a single correct trace.

Efficient RL and Reasoning. While reasoning LMs become more powerful, computation is more demanding during the training and deployment procedure. Therefore, researchers recently paid more attention to efficient reasoning in both directions. Pivotal Token Search (PTS) (Abdin et al., 2024) accelerates the Direct Preference Optimization (DPO) (Rafailov et al., 2023) training by identifying tokens in a language model generation that significantly impact the probability of success for the reasoning task. Following the memory efficiency but training time inefficiency introduced by GRPO (Shao et al., 2024; Guo et al., 2025), many follow-up works improve the training convergence speed, including DAPO (Yu et al., 2025), CPPO (Lin et al., 2025), PODS (Xu et al., 2025) and DUMP (Wang et al., 2025). (Team et al., 2025; Zhang et al., 2025; Aggarwal & Welleck, 2025) saves the token usage during inference by training with one or multiple levels of length penalty. Meanwhile, meta reinforcement fine-tuning (MRT) (Qu et al., 2025) makes inference token wasted less in meaningless reasoning steps by making the success rate steadily increase with the number of reasoning episodes. Compared with all of these previous works, we not only decrease the supervised signals during training by only introducing an expert solution when the current model cannot solve the current task with a probability relatively high enough, but guide the model with the expert hint to increase the RL training efficiency. which can provide a denser reward for speeding up the RL training procedure.

Related classical RL methods. DAgger (Ross et al., 2011) intermittently injects expert actions to correct agent behavior. Instead, BREAD adaptively inserts expert hints only when the agent fails and allows the model to complete the remaining of the reasoning trace, allowing for a natural curriculum. Go-Explore (Ecoffet et al., 2021) trains an agent that can solve all Atari games by branching from intermediate states, while BREAD branches out from expert traces. Methods like Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) and potential-based reward shaping (PBRS) (Ng et al., 1999) densifies learning signals under sparse rewards through goal relabeling or external shaping functions, but BREAD can achieve a similar effect through hint-based rollout initialization. Finally, while prior works like Kickstarting (Schmitt et al., 2018) and Expert Iteration (Anthony et al., 2017) explore teacher-student transfer via full trajectories or alternating control, BREAD explicitly focuses on sparse expert intervention with minimal demonstrations.

B. Motivation

B.1. Exploring and Contrasting SFT, RL, and BREAD under a Toy Model

It is known that a T times deeper LLM can internally simulate T chain-of-thought steps of a smaller LLM (Saunshi et al., 2025). This implies that the expert model can generate a dense reasoning trace which necessitates a $T\times$ longer simplified trace for the student model to digest via SFT. Recent work (Li et al., 2025) makes related empirical observation that SLMs can struggle to learn from strong reasoners. We propose modeling this phenomena through Markov chains as follows: Imagine a Markov chain with $K \times K$ transition matrix. The expert/strong model has access to the full transition matrix, while the student model are forbidden from implementing certain state transitions. Denote the learnable state transitions by $\mathcal{P} \subset [K] \times [K]$ where $[K] = \{1, 2, \ldots, K\}$ and consider the following navigation task:

Navigation Task: Start a trace from State 1. Obtain a reward of 1 upon reaching State K.

For simplicity, suppose the expert generates a deterministic path $[\alpha_0 = 1, \alpha_1, \alpha_2, \ldots, \alpha_{T-1}, \alpha_T = K]$ where $\alpha_t \in [K]$ for $0 \leq t \leq T$. During SFT, small model will only benefit from the expert model's trace when the expert transitions (α_t, α_{t+1}) are learnable, i.e., $(\alpha_t, \alpha_{t+1}) \in \mathcal{P}$. If no transition lies in $\mathcal{P}$ for $t = 0, 1, \ldots, T-1$, the small model cannot learn from SFT.

Without a good SFT-induced initialization, pure RL is known to suffer from sparse rewards (Vecerik et al., 2017). In the

context of the Navigation Task, further suppose that the small model can at most jump d steps away from the current state, i.e., $|j - i| \leq d$ for all $(j, i) \in \mathcal{P}$. This implies that we can only learn from expert trace when it makes small jumps i.e. $|\alpha_t - \alpha_{t+1}| \leq d$. Additionally, consider a poor-quality initialization where, at any state i, the small model has tendency to bounce back and forth the State i and its *favorable* neighbor $n(i)$ in the sense that all $j \neq n(i)$ obeys $\min\{\mathbb{P}(i \to j), \mathbb{P}(n(i) \to j)\} \leq \epsilon$ for some $\epsilon > 0$. Then, the initial model would achieve an exponentially-sparse reward proportional to $\epsilon^{\Omega(K/d)}$ under mild conditions on the maximum trace length.

To proceed, we have experimented with the Navigation Task controlled by the parameters number of states K, small model's jump capacity d, and initialization quality ϵ. Figure 5 demonstrates how optimization of SFT and GRPO can suffer as a function of n and ϵ respectively. Importantly, we also display the performance of BREAD which exhibits a much more favorable performance. The reader is referred to the supplementary material for full experimental details.

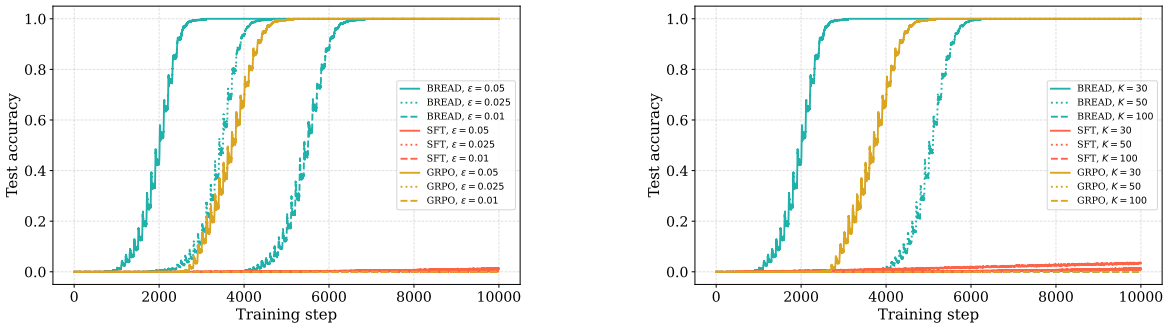
The BREAD is able to achieve sample efficiency by restricting the policy search to the unseen expert trace $[\alpha_s, \alpha_{s+1}, \ldots, \alpha_T = K]$. In the extreme case of $s = T - 1$, BREAD only finds a path from α_{T-1} to α_T which has substantially denser reward with exponent moving from $\Omega(\frac{K}{d})$ to $\Omega(\frac{K}{Td})$ assuming $|\alpha_t - \alpha_{t+1}| \propto K/T$. When $T \propto K$, the overall training time of BREAD is expected to be polynomial in K rather than exponential. In words, BREAD facilitates efficiency by learning from expert's reasoning one step at a time.

In the next section, we discuss how these insights are in line with the SFT and RL performance on real reasoning tasks with state-of-the-art models.

B.2. Empirical Insights into the Limitations of SFT and Reinforcement Learning for SLMs

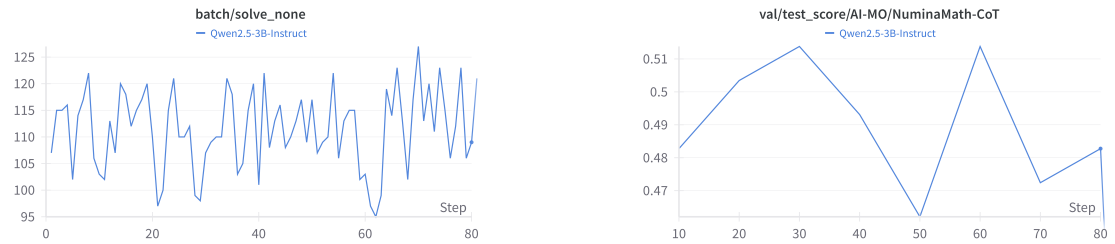
RL-only: We begin by assessing how well RL works with SLMs in isolation. Experiments with vanilla GRPO (Shao et al., 2024), displayed in Figure 6, show that it merely sharpens the capabilities the model already exhibits. GRPO relies on sampling a mix of good and bad traces. But when a small base model fails to produce any high-quality trace—***nearly half of the queries*** in our evaluations in Figure 6a—learning stalls due to lack of reward signals (Figure 6b). Related limitations are noted for RL with Verifiable Rewards of (Yue et al., 2025), which struggles to elicit fundamentally new reasoning patterns.

SFT-only: SFT can introduce new knowledge into the model. However, as discussed earlier, traces generated by much stronger models can be too complex for SLMs to learn resulting in poor initialization for RL. To demonstrate this, we start from Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct base models and fine-tune them with 1000 difficult questions, paired with reasoning traces generated by Gemini Thinking Experimental (Google, 2024) and Deepseek-R1 (Guo et al., 2025), following (Snell et al., 2024). These reasoning traces were previously shown to improve the reasoning capability of large models such as Qwen2.5-32B-Instruct and Qwen2.5-14B-Instruct by (Muennighoff et al., 2025). However, in our experiment results shown in Table 2, when SFT was performed on small models, ***accuracy actually decreases***. For example, accuracy dropped from 0.257 to 0.177 on the GPQA dataset when the 1.5B parameter model was fine-tuned on S1K traces, and dropped even further to 0.121 when fine-tuned on the more verbose S1K-1.1 data. This corroborates our central intuition: when traces used for SFT exceed an SLM's learning capacity, they can hurt the model performance rather than helping.



(a) Varying ϵ – Initialization difficulty (b) Varying K – Number of states

Figure 5. We compare SFT, GRPO, and BREAD according to the Navigation Task described in Section B.1. K is number of states in the Markov chain whereas ϵ is the probability of transition to *non-favorable* states. Our toy model reveals settings where BREAD can succeed while SFT or GRPO completely fail.



(a) Number of samples where all 8 rollouts fail

(b) Validation accuracy over training steps

Figure 6. Issues with vanilla GRPO on SLMs. Among 256 samples in a batch, for nearly half of them, the base model's 8 rollouts produce no correct trace. The absence of reward inhibits further performance gains and validation accuracy stalls.

These findings—that for small models, RL alone, or SFT + RL, are insufficient—motivate BREAD which uses expert traces directly in the RL phase instead of relying on SFT to learn them first.

Theoretical Model. Consider a Markov chain with a transition matrix P in a dimension of $2K \times 2K$ (Note that the assumption of even number of states does not affect the generalization of the final claims), where P is the ground-truth transition matrix. $P_{i,j} = 1 - \delta$ if and only if $i = j + 2 \mod 2K$. The expert traces start at some odd state $2i + 1$ and terminate once it visits all K odd states for the first time. This represents the correct reasoning traces provided by expert language models.

Student/Small Model. Student also has a $2K \times 2K$ Markov chain. However, since it is small, its parameters are restricted to be transitions between odd and even states. That is, we can only have $P_{2i,2j+1} \neq 0$. The success criteria is same: starting from some state $2i + 1$ initially, we want the student model to generate a trace that visits all K odd states. To make things tricky, we assume a poor initialization of $P_{2i,2i+1} = P_{2i+1,2i} = 1 - \varepsilon$ where ε is a small number. In this way, the chain has an initial tendency to bounce between $2i$ and $2i + 1$ back and forth, which simulates the small base models that cannot go through the hard crucial reasoning steps for a relatively complex question.

Based on the above theoretical model, we can have the following claims:

- **Claim 1:** SFT with the expert traces cannot help the student model learn anything. This is because the expert traces only contain transition between odd states while the student's transition probabilities between odd states are always 0.
- **Claim 2:** RL will take nearly forever to learn anything because of the bounce back and forth

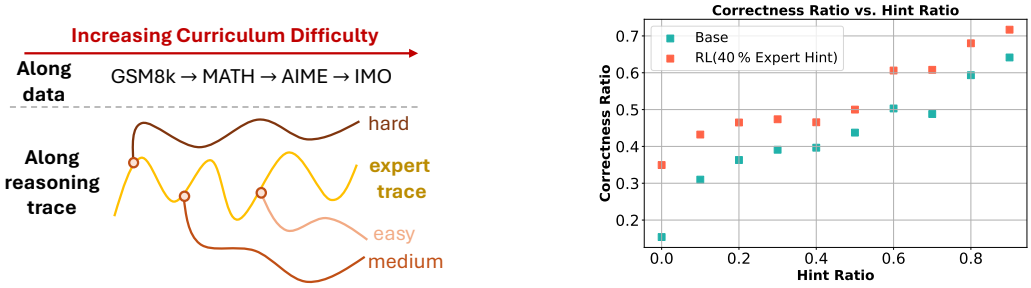
Dataset	Qwen-1.5B-Instruct			Qwen-3B-Instruct		
	Base	+ SFT (S1K)	+ SFT (S1K-1.1)	Base	+ SFT (S1K)	+ SFT (S1K-1.1)
MATH (Hendrycks et al., 2021)	0.494	0.426	0.408	0.624	0.616	0.630
GPQA (Rein et al., 2024)	0.258	0.177	0.121	0.369	0.247	0.288

Table 2. Accuracy of small models can decrease after SFT. The base SLMs are Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct and the datasets used for SFT are S1K and S1K-1.1 (Muennighoff et al., 2025), with responses generated by Gemini Flash Thinking API (Google, 2024) and DeepSeek-R1 (Guo et al., 2025) respectively.

B.3. Why BREAD: Expert Traces Provide Critical Guidance and Curriculum for RL

While Appendix B.2 shows that an SLM can struggle to imitate traces generated by powerful LLMs, we posit that ***the SLM can still understand them well enough to extract useful information***, namely by using part of the complex traces as hints to lower the problem difficulty. To illustrate this, we define the "hint ratio" as the fraction of the expert trace used (in terms of number of episodes), as illustrated in Figure 7a. A higher hint ratio means more guidance for the model. In Figure 7b, we plot the model accuracy for different hint ratios, where the hint is appended to the inference query. We can see that our intuition is correct: even providing part of the hint in a very simple way (by appending to the inference query) helps improve accuracy. In contrast, Figure 3 shows that SFT only on the same traces gets very limited improvement and can even hurt. BREAD is motivated by this observation and uses expert hints in a more sophisticated way, by integrating them directly into the RL create denser rewards.

We can also view BREAD through the lens of curriculum learning. As shown in Figure 7a, as we branch out from the expert trace earlier, solving the problem becomes more challenging. The Episode Anchor Search (EAS) step in BREAD



(a) Illustration of hint ratio (how much of the expert trace is provided) and curriculum learning.

(b) Accuracy for different ratio of hints, appended to the inference query.

Figure 7. (a) Math benchmarks may show difficulty tiers, but ranking individual problems for curriculum training is still challenging. Instead, BREAD can adaptively adjust the difficulty by automatically choosing the branching point. For easier questions, it would provide no or shorter hints (smaller hint ratio). (b) Model accuracy increases as longer hint is appended to the query. Dataset is the first 200 questions of NuminaMath-CoT. The blue dots are the base model (Qwen2.5-3B-Instruct), and the orange dots are RL finetuned using traces containing 40% hints.

automatically adjusts the branching point, producing a self-paced curriculum. In this sense, BREAD serves as a generalized curriculum-learning framework which (1) allows us to utilize SFT data within RL, (2) adapts the optimization to the problem difficulty, and (3) generates dense rewards by creating a curriculum along the reasoning trace. By branching along the expert trace at adaptive cut-points, BREAD automatically surfaces the most informative training samples and tunes their difficulty on the fly. This relieves us from hand-crafting a data-level schedule and lets the model discover its own curriculum.

A final surprising observation from Figure 7b is that conditioning RL training on questions with partial hints not only improves performance on hinted queries (e.g., orange dots with hint ratio 0.3 and above), but also improves accuracy on questions without any hints (i.e., orange dot with hint ratio of 0). This highlights the model's ability to generalize from partial short traces to full long traces.

The appendices are organized as follows:

- In Appendix C, we explain the calculation of estimated FLOPs (Appendix C.1) and additional details about the training and evaluation of BREAD (Appendix C.2).
- In Appendix D, we show that SFT further cannot improve the performance of models (Appendix D.1), BREAD can reduce the number of rollouts required during the sampling stage in training (Appendix D.2), and the distribution of the step number of expert solutions (Appendix D.3).
- In Appendix E, we explain the baseline `GRPO w/ Expert Trace` in detail and compare it to RL with an SFT loss.

C. Experiment details

C.1. FLOPs

We compute the estimate FLOPS following (Snell et al., 2024; Hoffmann et al., 2022; Sardana et al., 2023). The supervised finetuning, which include one forward and one backward phase, people use a common approximation $6ND$ (Hoffmann et al., 2022), and for inference, which include only one forward phase, people always use $2ND$ (Sardana et al., 2023). Here N represents model parameters, D is the total token count processed in that pass. So a forward phase takes $2ND$ while a backward phase take $4ND$. For estimation, we define the average length of a single question in one inference time as D_{sample}, so $D = D_{\text{sample}} \times n_{\text{rollout}}$ for RL and $D = D_{\text{sample}}$ for SFT. For estimation, we define the average length of a single question in one inference time as D_{sample}, so $D = D_{\text{sample}} \times n_{\text{rollout}}$ for RL and $D = D_{\text{sample}}$ for SFT. GRPO with eight rollouts, $n_{\text{rollout}} = 8$ including eight forward and eight backward phase needs $6 \times 8 \times ND_{\text{rollout}}$. BREAD will take more forward pass to do the binary search, so we use $6 * 8 * ND_{\text{rollout}} + 4ND_{\text{additional}}$. `GRPO w/ Expert trace` includes including eight forward and nine backward phase, so we use $6 \times 8 \times ND_{\text{rollout}} + 4 \times ND_{\text{rollout}}$. We estimate D with the average length of all expert trace in the training dataset. We truly record the additional number of generation $D_{\text{additional}} = D_{\text{rollout}} \times n_{\text{additional\_rollout}}$. Notably, the generation length of BREAD is always shorter than expert trace and vanilla GRPO. So our BREAD can reduce even more computation resource compare to 75% shown in Figure 3b.

Algorithm 1 BREAD: GRPO with Branch Rollouts and Expert Anchors (full version)

Require: Dataset $\mathcal{D}$, current policy π_θ, sampling number G, a list of keywords for splitting episodes $[w_0, w_2, ..., w_J]$

procedure BREAD($\mathcal{D}, \pi_\theta, G, [w_0, w_2, ..., w_J]$)
  for step $= 1, 2, ..., N$ **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Update the old policy model $\pi_{\text{old}} \leftarrow \pi_\theta$
    for each question and expert solution pair (Q, S) in $\mathcal{D}_b$ **do**
      $[R_1, R_2, ..., R_G], \rho \leftarrow \texttt{EAS}(Q, S, \pi_\theta, G, [w_0, w_2, ..., w_J])$ ▷ Episode Anchor Search (EAS)
      Add the $(Q, \rho), R_i$ to the buffer.
    end for
    For each $(Q, \rho), o_i$ in the buffer, compute $\hat{A}_{i,t}$ for the t-th token of o_i ▷ Equation (2)
  end for
  for iteration $= 1, 2, ..., T$ **do**
    Update the policy model π_θ by maxmizing the BREAD objective ▷ Equation (1)
  end for
  return π_θ
end procedure
procedure EAS($Q, S, \pi, G, [w_0, w_2, ..., w_J]$)
  $[R_1, R_2, ..., R_G] \leftarrow \pi(Q)$ ▷ Sample I responses for question Q with current policy π
  $p_{\text{correct}} \leftarrow \texttt{compute\_correct\_probability}([R_1, R_2, ..., R_G], S)$
  ▷ compute correct probability based on responses and the gold solution
  if $0 < p_{\text{correct}}$ **then**
    return $[R_1, R_2, ..., R_G], NA$
  else
    $[e_0, e_1, ..., e_K] \leftarrow \texttt{split\_episodes}(S, [w_0, w_2, ..., w_J])$
    return BINARY_SEARCH_AND_GENERATE$(Q, S, \pi, G, [e_1, e_2, ..., e_K], 0, K)$
  end if
end procedure
procedure BINARY_SEARCH_AND_GENERATE$(Q, S, \pi, G, [e_1, e_2, ..., e_K], L, R)$
  $M = \frac{L+R}{2}$
  $[R_1, R_2, ..., R_G] \leftarrow \pi([Q, e_1, e_2, ..., e_M])$
  $p_{\text{correct}} \leftarrow \texttt{compute\_correct\_probability}([R_1, R_2, ..., R_G], S)$
  if $0 < p_{\text{correct}} < 1$ **then**
    return $[R_1, R_2, ..., R_G], S_{1:M}$
  else if $p_{\text{correct}} = 0$ **then**
    $L = M, R = R, M = \frac{L+R}{2}$
    return binary_search_and_generate$(Q, S, \pi, G, [e_1, e_2, ..., e_K], L, R)$
  else
    $L = L, R = M, M = \frac{L+R}{2}$
    return binary_search_and_generate$(Q, S, \pi, G, [e_1, e_2, ..., e_K], L, R)$
  end if
end procedure

C.2. Experiment Setting Details for BREAD

In addition to Section 3, we list additional details of our experiments here. During training, we set `max_prompt_length` = 2048 for MATH experiments and `max_prompt_length` = 4096 for NuminaMath-CoT experiments. For both training and evaluation, we set `max_response_length` = 4096 for MATH experiments and `max_response_length` = 8192 for NuminaMath-CoT experiments. During training, we set the number of rollouts as 8. We used a low-variance KL divergence and set the coefficient for KL divergence as 0.001. We set the temperature at 0.6 for both training and evaluation.

For the implementation details of the episode splitting for the expert trace during training, we split the expert trace with sentences separated by ``. '' or ``\n'' for MATH and paragraphs separated by ``\n\n'' for NuminaMath-CoT. Note that the expert traces can be either human provided solutions or correct solutions provided by larger expert models, and the splitting method here can also be changed to split according to a specific keyword list. After splitting, we aggregate the traces into 10 episodes evenly for all expert traces. The reason why we implement this way is that we want to make sure that the number of episodes of the expert traces is not too large, which can guarantee that the `EAS` step does not take too much time.

During training, assume we have a question Q and an expert hint ρ (or without ρ during inference), the template of the prompt is as follows:

{'content': '<|im_start|>system\nYou are a helpful assistant. You first thinks about the reasoning process in the mind and then provides the user with the answer.<|im_end|>\n<|im_start|>user\n$\{Q, \rho\}$ Show your work in <think> <\think> tags. And return the final answer within \\boxed{}.<|im_end|>\n<|im_start|>assistant\nLet me solve this step by step.\n<think>', 'role': 'user'}

For the hardware requirements, all of experiments are done with 8 L40S 40GB GPUs except the training starting from Qwen2.5-3B-Instruct as the base model, which requires 8 80G H100 GPUs.

D. Additional Experiments

D.1. SFT further no help

Supplement to Figure 3a. As shown in Figure 8, training SFT for more iterations does not further improve test accuracy. In other words, we already use the model that SFT can achieve. For a fair comparison, we therefore use the 300-step checkpoint as the starting point for `SFT+GRPO`.

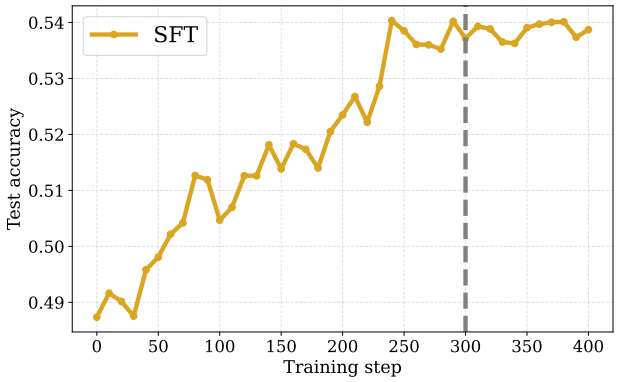


Figure 8. Test accuracy of SFT over training steps. The accuracy doesn't continuously increase after the number of training step we use.

D.2. BREAD can reduce number of rollout needed

Another straightforward way to reduce training cost is to reduce the number of rollouts, since FLOPs scale linearly with that count. However, for vanilla GRPO, fewer rollouts lead to lower accuracy, whereas BREAD maintains its performance even as the number of rollouts decreases. The result is shown in Figure 9. As we have shown in the main body, when the

task is so difficult that the base policy rarely produces correct traces, vanilla GRPO fails. To study rollout efficiency under conditions where GRPO can learn, we moved to the MATH dataset with a Qwen-2.5-3B-Instruct base model. With eight rollouts, GRPO does increase accuracy. However, performance will decrease if the rollout budget is reduced from 8 to 5. In contrast, BREAD reaches the same accuracy with just five rollouts, cutting training FLOPs while preserving performance.

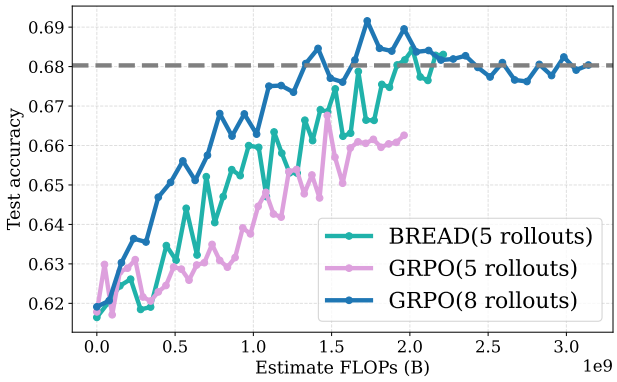
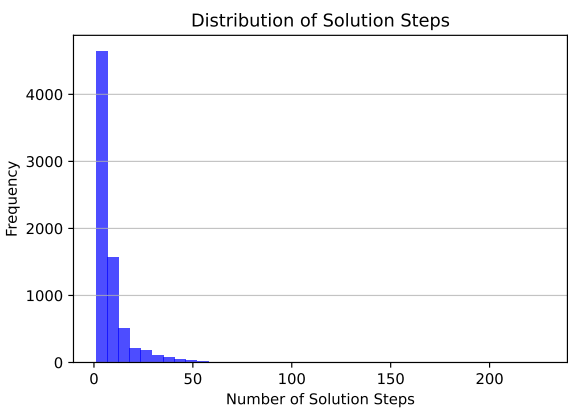


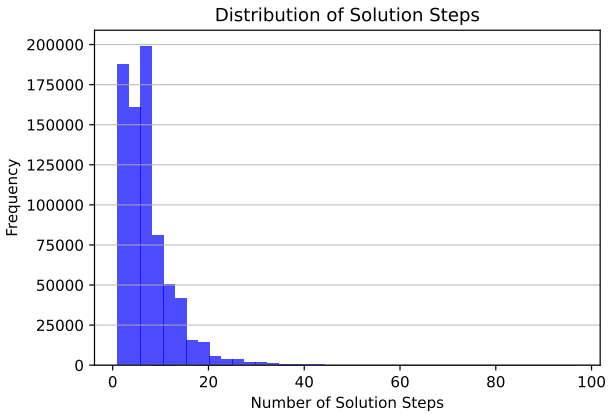
Figure 9. Test accuracy over training steps with different number of rollouts. The gray dashed line shows the final accuracy of vanilla GRPO with 8 rollouts. The total training step is 500.

D.3. Episode details

As described in Appendix C.2, we plot the distribution of the number of steps for our two datasets before episode aggregation. As shown in Figure 10, we can see that that most expert traces contain less than 20 steps, while few of them contain much more steps, which may slow down the training sif there ised if there is no episode aggregation.



(a) Distribution plot of MATH solution step numbers.



(b) Distribution plot of NuminaMath-CoT solution step numbers.

Figure 10. Distribution plot of MATH and NuminaMath-CoT solution step numbers

E. `GRPO w/ Expert Trace` Details and its Connection with RL with SFT Loss

The baseline `GRPO w/ Expert Trace` (GRPO-ET) generally follows the same procedure as the standard GRPO. Instead, it enforces one of the G rollouts as the expert trace. Therefore, the objective function is

$$\begin{aligned}\mathcal{J}_{\text{GRPO-ET}}(\theta) = \mathbb{E}_{(q,S,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G-1}\sim\pi_{\text{old}}(\cdot|q)} \\ \left[\frac{1}{G}\left(\sum_{i=1}^{G-1}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right)\right)\right] \\ + \frac{1}{G|o_S|}\sum_{t=1}^{|o_S|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{S,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{S,t}\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right) \end{aligned} \tag{3}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i<t})}{\pi_{\text{old}}(o_{i,t}|q,o_{i<t})}, \quad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G-1}, R_S)}{\text{std}(\{R_i\}_{i=1}^{G-1}, R_S)}, \quad \hat{A}_{S,t} = \frac{R_S - \text{mean}(\{R_i\}_{i=1}^{G-1}, R_S)}{\text{std}(\{R_i\}_{i=1}^{G-1}, R_S)} \tag{4}$$

All notations in Equations (3) and (4) are the same as (Guo et al., 2025), while S represents the expert trace, and all variables whose subscript contains S represent the corresponding variables.

Here, we can see a clear connection between `GRPO w/ Expert Trace` and GRPO (rollout number is $G-1$) with an SFT loss. Suppose that we want to deploy GRPO while adding the SFT entropy loss to the standard GRPO loss, the objective function is

$$\begin{aligned}\mathcal{J}_{\text{GRPO\_SFT}} = \mathbb{E}_{(q,S,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G-1}\sim\pi_{\text{old}}(\cdot|q)} \\ \left[\frac{1}{G}\left(\sum_{i=1}^{G-1}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right)\right)\right] \\ + \sum_{t=1}^{|o_S|}\log\pi_\theta(S_t \mid q, S_{<t}) \end{aligned} \tag{5}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i<t})}{\pi_{\text{old}}(o_{i,t}|q,o_{i<t})}, \quad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G-1}, R_S)}{\text{std}(\{R_i\}_{i=1}^{G-1}, R_S)} \tag{6}$$

There are 3 main difference between Equation (3) and Equation (5):

1. A coefficient for the expert trace loss $\frac{1}{G|o_S|}$
2. A KL divergence term $-\beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$
3. An Advantage term $\min\left(r_{i,t}(\theta)\hat{A}_{S,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{S,t}\right)$

Suppose that $r_{i,t} \leq 1+\varepsilon$, because $\hat{A}_{S,t} \geq 0$, $\min\left(r_{i,t}(\theta)\hat{A}_{S,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{S,t}\right) = r_{i,t}(\theta)\hat{A}_{S,t}$, which is exactly the entropy loss with a coefficient $\hat{A}_{S,t}$ if we replace $\pi_{\text{old}}(o_{i,t}|q,o_{i<t})$ in $r_{i,t}(\theta)$ with the one hot embedding of the expert trace tokens. Therefore, `GRPO w/ Expert Trace` can not only have better training consistency, but can also assign different credits according to the token probability ratio between the old and the current policy.

F. Additional Details and Analysis of the Markov Model

In this section, we present a detailed and precise formulation of the Markov model introduced in Section B.1.

Recall that we consider a Markov chain with K states (indexed by $1, 2, \cdots, K$) and a $K \times K$ transition matrix. We assume that, for the expert/large model, all $(i \to j)$ transitions are learnable in the transition matrix. However, for a small/student

Algorithm 2 GRPO w/ Expert Trace (GRPO-ET)

Require: Dataset $\mathcal{D}$, current policy π_θ, sampling number G
procedure GRPO_WITH_EXPERT_TRACE($\mathcal{D}, \pi_\theta, G$)
  for step $= 1, 2, ..., N$ **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Update the old policy model $\pi_{\text{old}} \leftarrow \pi_\theta$
    Sample $G-1$ outputs $\{o_i\}_{i=1}^{G-1} \sim \pi_{\text{old}}(\cdot|Q)$ for each question $Q \in \mathcal{D}_b$
    Combine the expert trace with the sampled outputs to construct $\left\{\{o_i\}_{i=1}^{G-1}, S\right\}$
    Compute rewards $\left\{\{r_i\}_{i=1}^{G-1}, r_S\right\}$ for each sampled output o_i and expert trace S
    ▷ r_S is generally 1 because of solution correctness
    For each o_i and expert trace S. compute $\hat{A}_{i,t}$ for the t-th token of o_i and S. ▷ Equation (4)
    for iteration = $1, ..., T$ **do**
      Update the policy model π_θ by maximizing the GRPO-ET objective ▷ Equation (3)
    end for
  end for
  return π_θ
end procedure

model, not all $(i \to j)$ transitions are learnable and we denote the learnable state transitions by $\mathcal{P} \subset [K] \times [K]$ where $[K] = \{1, 2, \ldots, K\}$.

We first introduce the following definition of pretrained small model utilized in our experiments.

Definition F.1. Pretrained Small Model A pretrained small model is defined as a Markov model $\mathcal{M} = (\mathcal{S}, \boldsymbol{P})$, where $\mathcal{S} = \{1, 2, \ldots, K\}$ is the state space and $\boldsymbol{P} \in \mathbb{R}^{K \times K}$ is the transition matrix. The model satisfies the following properties:

- Let $d \geq 1$ denote the maximum allowed jump between states. The set of learnable state transitions $\mathcal{P}$ satisfies:

$$(i, j) \in \mathcal{P} \quad \text{if and only if} \quad |i - j| \leq d.$$

- For some $\epsilon \ll 1/d$, the transition probabilities satisfy:

$$\boldsymbol{P}_{ij} := \mathbb{P}(i \to j) = \begin{cases} 1 - \Theta(d\epsilon), & \text{if } i = j, \\ \Theta(\epsilon), & \text{if } (i, j) \in \mathcal{P} \text{ and } i \neq j, \\ 0, & \text{if } (i, j) \notin \mathcal{P}. \end{cases}$$

Our goal is to finetune a small model defined in Definition F.1 to solve the following task:

Navigation Task: Start a trace from State 1. Obtain a reward of 1 upon reaching State K.

Note that the number of states K, the weak transition probability ϵ, and the maximal allowed jump distance d control the difficulty of the task.

Theorem F.2. *Suppose the maximum trace length satisfies $T_{max} = \Theta(K/d)$, and consider the navigation task where a trajectory starts from State 1 and receives a reward of 1 only upon reaching State K. Then, a small model defined in Definition F.1 achieves an expected reward of $\epsilon^{\Theta(K/d)}$.*

The theorem highlights a key limitation of applying standard reinforcement learning algorithms (e.g., GRPO) to the navigation task: since these methods rely on observing non-zero reward trajectories to propagate gradients and update model parameters, they are highly inefficient in hard settings. Specifically, in the small model regime, the agent must generate approximately $\epsilon^{-\Theta(K/d)}$ trajectories before observing a single successful episode that reaches State K. This exponential sample complexity causes a significant challenge for learning.

In the following section, we introduce the BREAD algorithm of the Markov model that leverages SFT trajectories to improve sample efficiency.

Definition F.3. An SFT trajectory is a sequence $[\alpha_0, \alpha_1, \cdots, \alpha_T]$ where $\alpha_i \in [K]$ and $\alpha_0 = 1 \leq \alpha_{i-1} \leq \alpha_i \leq \alpha_T = K$ for $i \in [T]$. Define the maximal jump distance of the SFT trajectory by

$$D := \max_{i \in [T]} |a_i - a_{i-1}|.$$

The trajectory is said to be infeasible for a small model defined in Definition F.1 with maximum jump distance d if it contains at least one transition exceeding the allowed jump size. That is $D > d$.

Since any infeasible SFT trajectory (cf. Definition F.3) contains at least one transition $(\alpha_{i-1} \to \alpha_i) \notin \mathcal{P}$, it is evident that the small model (cf. Definition F.1) cannot learn directly from such a trajectory, as the corresponding transitions within the trajectory lie outside its learnable set $\mathcal{P}$.

Algorithm 3 BREAD_Markov

Require: An SFT trajectory $[\alpha_0, \cdots, \alpha_T]$, initial small model $\mathcal{M}([K], \boldsymbol{P})$, reward threshold r_{thred}, maximal trajectory length T_{max}
1: **for** episode $t = T, T-1, \ldots, 1$ **do**
2: $r \leftarrow 0$
3: **while** $r < r_{\text{thred}}$ **do**
4: Sample N trajectories from $\mathcal{M}$ starting from state α_t with maximal length $T_{\text{max}} - t$
5: Update the current reward: $r \leftarrow r_{\text{new}}$
6: Update the transition matrix: $\boldsymbol{P} \leftarrow \boldsymbol{P}_{\text{new}}$
7: **end while**
8: **end for**

Theorem F.4. *Let an SFT trajectory (cf. Definition F.3) has maximal jump distance $D = \Theta(K/T)$, and suppose the maximum trace length satisfies $T_{max} = T + \Theta(K/dT)$. Then, by finetuning a small model (cf. Definition F.1) using BREAD algorithm as described in Algorithm 3, the model obtains an expected reward of $\epsilon^{\Theta(K/Td)}$ at each iteration.*

Theorem F.4 demonstrates that compared to the pure RL approach in Theorem F.2, the BREAD algorithm achieves substantially denser rewards. Specifically, the exponent improves from $\Theta(\frac{K}{d})$ to $\Theta(\frac{K}{Td})$, while using a shorter trace length. Intuitively, BREAD improves sample efficiency by decomposing the expert's reasoning into smaller steps, allowing the small model to acquire knowledge one step at a time.

Experimental settings for Figure 5: We conduct experiments by finetuning a Markov model (cf. Definition F.1) using three different methods: SFT, GRPO, and BREAD. Note that in our experiments, to introduce additional randomness, for each state $i \in [K]$, we randomly select one over its connected states (e.g., among $i-d, \ldots, i, i+d$ states) and assign it the highest transition probability of $1 - \Theta(\epsilon)$, instead of always assigning the highest probability to $\mathbb{P}(i \to i)$. In both Figs. 5a and 5b, $d = 2, T_{\text{max}} = 2K$ and SFT is infeasible (cf. Definition F.3) with length $T = K/2$. In Fig. 5a, we fix $K = 30$ and vary $\epsilon \in \{0.01, 0.025, 0.05\}$. In contrast, in Fig. 5b, the $\epsilon = 0.05$ is remained unchanged and the number of states varies in $K \in \{30, 50, 100\}$. Each experiments is trained for 10000 iterations with 1000 trajectories sampled with each iteration.

F.1. Proof of Theorems F.2&F.4

In this section, we provide the following theorem along with its proof. Theorems F.2 and F.4 can be directly derived as its corollaries.

Theorem F.5. *Suppose the maximum trace length satisfies $T_{max} = \Theta(m)$, and consider the navigation task where a trajectory starts from State i and receives a reward of 1 only upon reaching State $j := (i + md)$. Then, a small model defined in Definition F.1 achieves an expected reward of $\epsilon^{\Theta(m)}$.*

Proof. Given this navigation task where a reward of 1 is received upon reaching the goal state $j := i + md$, the expected can be interpreted as the probability of successfully reaching the final goal with T_{max} steps. Define the successfully reaching probability by $\mathbb{P}(i \to j; T_{\text{max}})$. In the following, we derive both lower and upper bounds of this probability.

• **Lower bound:** The lower bound can be easily derived by considering a single successful trajectory, presented as follows

$$(i) \to (i+d) \to (i+2d) \to \cdots \to (i+md).$$

Given that the probability of each step is $\Theta(\epsilon)$ following Definition F.1 and there are m steps required, the probability of obtaining such trajectory is $\Theta(\epsilon^m)$. It serves as the lower bound of the successfully reaching probability (assuming $m < T_{\text{max}}$) and we have

$$\mathbb{P}(i \to j; T_{\text{max}}) \geq \Theta(\epsilon^m).$$

• **Upper bound:** To obtain the upper bound, we first consider a moving-forward-only Markov chain. That is, the transition matrix $\tilde{\boldsymbol{P}} \in \mathbb{R}^{K\times K}$ is defined as follows:

$$\tilde{\boldsymbol{P}}_{ij} := \tilde{\mathbb{P}}(i \to j) = \begin{cases} 1 - \Theta(d\epsilon), & \text{if } i = j, \\ \Theta(d\epsilon), & \text{if } j = i + d, \\ 0, & \text{otherwise.} \end{cases}$$

This setup restricts the agent to either stay in the current state or move forward exactly d steps with each transition. Note that $\tilde{\boldsymbol{P}}$ has strictly higher successfully reaching probability compared to $\boldsymbol{P}$ in Definition F.1. Therefore, we focus on upper bounding the reaching probability of the simplified forward-only model, denoted by $\tilde{\mathbb{P}}(i \to j; T_{\max})$.

Given the maximal trace length T_{max}, at each timestep, the agent can either stay in the current state or move forward to the state that is d steps away. Then as long as at least m out of the total T_{max} are forward transitions, the agent will reach the final goal state within T_{max} steps. Therefore, we have

$$\begin{aligned} \tilde{\mathbb{P}}(i \to j; T_{\text{max}}) &= \sum_{m'=m}^{T_{\text{max}}} \binom{T_{\text{max}}}{m'} \Theta(d\epsilon)^{m'} (1 - \Theta(d\epsilon))^{T_{\text{max}} - m'} \\ &\le 2^{T_{\text{max}}} \Theta(d\epsilon)^{m}. \end{aligned}$$

Given that $T_{\max} = \Theta(m)$, we get

$$\mathbb{P}(i \to j; T_{\text{max}}) \le \tilde{\mathbb{P}}(i \to j; T_{\text{max}}) \le \Theta(d\epsilon^{m}).$$

Combining lower and upper bound, and considering a constant jump distance $d \ll \epsilon^{-1}$, it completes the proof. □

G. Discussion and Limitations

Our work introduces BREAD as a principled algorithm to densify the reward signal using branch rollouts from the expert trace. BREAD significantly enhances sample complexity, training time, and eventual accuracy over employing GRPO. It is also theoretically well motivated and allows the model to overcome fundamental bottlenecks of SFT+GRPO. BREAD has also a few limitations: Firstly, we focus on SLMs and assume that there is a strong expert/teacher model to provide high-quality traces. Secondly, it is possible that SLM may fail to obtain a reward signal even from partial traces of the teacher-imagine the student model can't conclude even if the full proof is presented. We leave these as future directions.

Broader impact: Enhancing the capabilities of small language models can have environmental benefits by improving AI efficiency. We do not anticipate negative societal impacts.