e3: Learning to Explore Enables Extrapolation of Test-Time Compute for LLMs

Amrith Setlur [*1] **Matthew Y. R. Yang** [*1] **Charlie Snell** [2] **Jeremy Greer** [3] **Ian Wu** [1] **Virginia Smith** [1] **Max Simchowitz** [1] **Aviral Kumar** [1]

Abstract

Test-time scaling offers a promising path to improve LLM reasoning ; however, the true promise of this paradigm lies in *extrapolation* (*i.e.,* to scale performance as LLMs "think" for longer). We show that one way to enable extrapolation is by training the LLM at *in-context exploration*; that is, training the LLM to effectively spend its test time budget by chaining operations (such as generation, verification, refinement, *etc.*). To enable in-context exploration, we identify three key ingredients as part of our recipe `e3`: (1) chaining asymmetries in base LLM competence, *e.g.*, chaining verification (easy) with generation (hard), as a way to implement in-context search; (2) leveraging negative gradients from incorrect traces to amplify exploration that chains additional asymmetries ; and (3) aligning task difficulty with training token budget to structure in-context exploration. Our recipe `e3` produces the best performing 1.7B model on AIME/HMMT'25, and can also extrapolate compute to 2.5× the model training budget.

1. Introduction

Many recent works post-train LLMs via reinforcement learning (RL) (DeepSeek-AI et al., 2025; Yu et al., 2025) and supervised fine-tuning (SFT) (Team, 2025; Muennighoff et al., 2025) at long context windows. However, it is unclear whether the models post-trained with current recipes can truly realize the promise of ***extrapolation*** (see App. B): if we scale the test compute beyond the *training budget*, would the LLM be able to continue to solve more problems?

In this paper, we show that the key to enabling extrapolation is ***learning to explore in-context***: if a model learns to use compute by searching through multiple reasoning paths or implementing algorithmic procedures, it can "guide" the search towards the correct answer, and improve its performance with more test compute. To demonstrate this, we build a recipe `e3` based on following ingredients:

1) Asymmetries are critical for learning to explore. In the absence of external tools, we show that feedback can emerge from ***asymmetries***, differences in the model's competence at different procedures that constitute an output trace. One example is the verification-generation (VG) gap, where models are more capable of verifying their answers than generating correct ones. While prior work (Setlur et al., 2025; Swamy et al., 2025; Song et al., 2024; Kim et al., 2025; Gandhi et al., 2025) has noticed such asymmetries, we show that these are critical for extrapolation, meaning that in their absence, scaling is strikingly hard.

2) Negative gradient in RL amplifies in-context exploration. If asymmetries are a prerequisite for learning to explore, what enables them to evolve and facilitate learning useful exploration strategies during RL training? We show that ***negative gradients*** (*i.e.*, gradients on incorrect traces, see App. B) is a key enabler of in-context exploration, when the base model presents asymmetries. Negative gradients drive exploration (Tajwar et al., ICML 2024; Ren & Sutherland, 2024) by moving the probability mass from shorter failed traces onto longer traces that *"chain"* new asymmetries (*e.g.*, verifying a calculation once more).

3) Structured exploration with coupled curriculum. While negative gradients amplify asymmetries and produce longer responses, at larger budgets, RL often suffers from poor training convergence (Agarwal et al., 2021). While one could train with a smaller budget, we show that training on hard problems at short budgets often disincentivizes exploration since the model is forced to commit to an answer prematurely. To resolve this, we design a ***coupled curriculum*** over pairs of (data mixture, training budget) that effectively structures exploration driven by the negative gradient.

The above constitutes our recipe `e3`, which we use to post-train Qwen3-1.7B with a training budget of 16k. We achieve the **best performance at <2B scale on AIME'25 and HMMT'25**, and our model's performance consistently im-

*Equal contribution [1]Carnegie Mellon University [2]UC Berkeley [3]Oumi. Correspondence to: <asetlur@andrew.cmu.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Project website: https://matthewyryang.github.io/e3/.

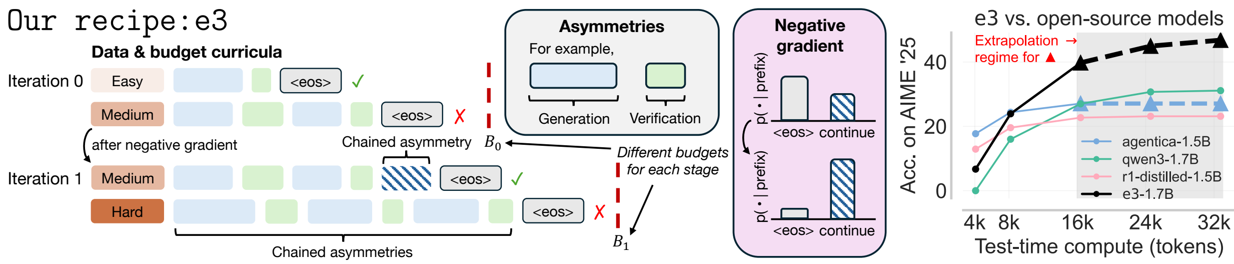


Figure 1: ***In-context exploration enables extrapolation*** **(e3)**: (i) chaining asymmetric capabilities in the base model, *e.g.,* reliably self-verifying responses after generating them; (ii) using negative gradients in RL training to penalize incorrectly terminated model responses, lengthening them further with more chained asymmetries, until the correct answer is discovered; and (iii) data & budget curricula for RL training that carefully balances explore-exploit tradeoff by sequentially training models on different datasets and training compute budgets. Qwen3-1.7B fine-tuned with e3 extrapolates test-time compute outperforming all ≤2B models on AIME'25.

proves as we extrapolate the test-time budget to 40k. Please refer to Appendix A for a discussion of related works.

2. Asymmetries in the Base Model: A Prerequisite for In-Context Exploration

In this section, we demonstrate that when the base model exhibits *asymmetric* competence at different skills, RL post-training prefers to learn solutions that ***chains asymmetric skills*** in ways that improve final performance. We focus on a key special case when the model is more accurate at verifying its own answers than it is at generating correct ones; that is, when the model exhibits a *verification-generation gap* (***VG Gap***), on a particular problem domain (Song et al., 2024). We show that RL training on problem domains with VG gap (i) encourages chaining asymmetries, (ii) enables in-context exploration that (iii) discovers new solutions, often extrapolating to larger budgets and OOD problems.

Definition 2.1 (Chaining asymmetric capabilities p, q in model π.)**.** Let $p, q : \mathcal{S} \mapsto \mathcal{S}$ be functions over token sequences $\mathcal{S}$ (*e.g.*, p can be generation, q can be verification), and $\texttt{detect}(f, \tau)$ detects number of calls to function f in a token trace τ. For a reward r, we say that policy π chains asymmetries p, q if it benefits from calls to the composition $q(p(\cdot))$, compared to only $p(\cdot)$:

$$\mathbb{E}_{\tau\sim\pi}\left[r(\tau) \mid \texttt{detect}(q(p(\cdot)), \tau) > 0\right] > \mathbb{E}_{\tau\sim\pi}\left[r(\tau) \mid \texttt{detect}(p, \tau) > 0\right],$$

even though there is an optimal policy $\pi^\star_r$ that never calls q, i.e., $\mathbb{E}_{\tau\sim\pi^\star_r}\texttt{detect}\left[(q, \tau)\right] = 0$.

Setup. We validate the role of asymmetries in learning to explore by investigating two didactic tasks, on which Llama3.2-3B admits different VG gaps. First, the Countdown game (Yao et al., 2023; Gandhi et al., 2024) (CDOWN) requires converting a set of numbers into an equation that outputs the desired target. T Second, we study n-digit multiplication MULT where the base model exhibits limited verification (see App. E for asymmetry gap on MULT). Additionally, we supervise fine-tune Llama3.2-3B on correct n-digit multiplication traces containing verification to encourage more verification attempts (MULT-V). MULT vs. MULT-V evaluates the presence of asymmetries in base LLM.

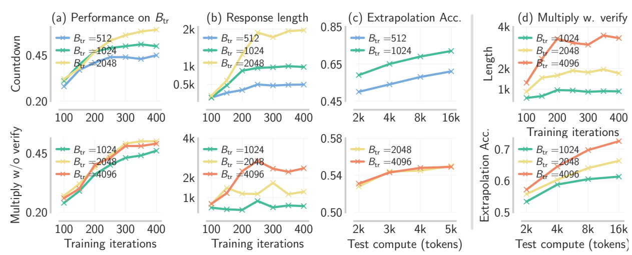


Figure 2: **RL training with and w/o. asymmetries in** π_b**.** When asymmetries (*e.g.,* VG gap) are present (*e.g.,* in CDOWN), RL training amplifies response length by chaining more asymmetries to explore in-context On the other hand, when VG gap is absent in π_b (*e.g.,* in MULT), increases in length and extrapolation performance are subdued. When we explicitly train on a base model fine-tuned to verify MULT (referred to as the MULT-V), we again observe upward length and extrapolation trends, consistent with CDOWN.

1) Verification-generation asymmetry in π_b ***improves the performance of RL trained solutions.*** Fig. 2(a,b) shows a stark difference in performance and response length as we vary B_{tr} on CDOWN and MULT . On CDOWN, performance consistently increases as B_{tr} increases from 512 → 2048, accompanied by a clear increase in length. On MULT, where the base model has limited propensity to verify, performance increases when B_{tr} increases from 1024 to 2048, but plateaus thereon. Contrast this with Fig. 2(d), RL training on MULT-V , which exhibits longer lengths and stronger extrapolation performance because it leverages asymmetries. Therefore, **asymmetries improve performance and length-utilization in RL post-training.**

2) Chaining asymmetries enable extrapolation via in-context exploration. Interleaving verification and generation steps chains together asymmetric capabilities of the base model; we refer to this as ***chaining asymmetries***. In Fig. 2 (c), we plot the extrapolation performance of the models trained at two values of B_{tr}. On CDOWN the model trained with B_{tr}=0.5/1k makes steady progress on problems in test budgets that are 8-16× B_{tr} itself. On MULT we find that B_{tr} has no effect on extrapolation performance when the base LLM does not have asymmetries, but has a substantial effect when asymmetries are present. More importantly, while the base model without VG asymmetry fails to extrapolate and solve unsolved problems, with its accuracy improving by merely $\leq 2\%$ despite 16x compute scaling, the base model with VG asymmetry can still extrapolate well. See App. C for a theoretical model that explains why asymmetries enable exploration. In App. E, we discuss the performance impact of chained asymmetries.

3. Negative Gradients Incentivize Exploration that Chains Asymmetries

Having observed that the presence of asymmetry in the base model is a prerequisite for in-context exploration, the next question is: What enables models to exploit these asymmetries during RL? In this section, we show that a key ingredient is the ***negative gradient***, the gradient term multiplied by a negative advantage in the standard GRPO / PPO objective (see App. B). Negative gradient drives in-context exploration via two mechanisms: (i) incentivizing the sampling of unseen token sequences; (ii) chaining asymmetries like VG gap (Sec. 2) that rapidly drives up response length. For brevity, we refer to (i) as "exploration" (Amin et al., 2021) and (2) as in-context exploration (or "meta-exploration" (Duan et al., 2016; Gupta et al., 2018)).

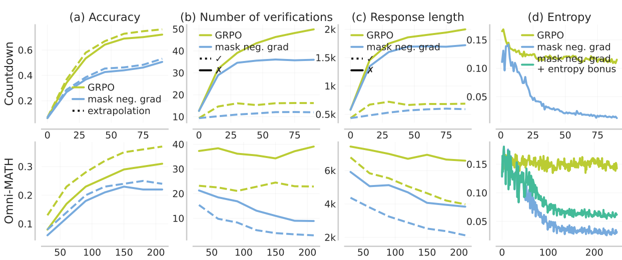


Figure 3: ***RL training with and w/o. negative gradients:*** When the base model presents asymmetries, negative gradients promote in-context exploration by: (i) increasing length (shown in (c)) and chaining more asymmetries on incorrect responses (shows up as more verification attempts (b)), and (ii) increasing entropy and thus esponse diversity (d). This leads to better performance on both training and extrapolation budgets. In (b, c), ✓ denotes the statistic computed on correct responses and ✗ on incorrect ones.

Analysis setup. We analyze the evolution of response length, performance, and the number of chained asymmetries of two training algorithms: (i) standard GRPO (Shao et al., 2024) with token-level normalization (Yu et al., 2025); (ii) GRPOMask, which zeros out (i.e. masks) the negative gradient and whilst retaining the *positive* gradient, resembling online STaR (Zelikman et al., 2022) or RFT (Yuan et al., 2023). We conduct our experiments on CDOWN and DMATH reasoning (from the DeepScaleR dataset (Luo et al., 2025b)). We make the following observations:

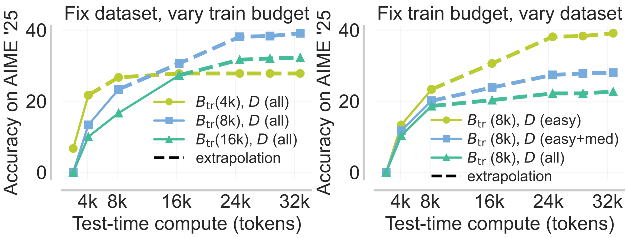
1) Negative gradient increases the number of chained asymmetries, and thereby boosts meta-exploration. When applied on an incorrect response $\mathbf{y}$ with tokens $y_1, y_2, .., \texttt{EOS}$, negative gradient reduces $p(y_i|\mathbf{y}_{1:i-1})$, including $p(\texttt{EOS}|\mathbf{y})$ when the response ends before the training budget. Fig. 3(b) reveals that the probability mass recovered from the negative gradient (note: total probability is conserved) is repurposed to increase the probability of chaining new pairs of asymmetric skills to the current trace (*e.g.*, "Wait, ..." instead of terminating with EOS). When negative gradients are masked (GRPOMask) in CDOWN, we see that attempts (b) and length (c) plateau, accompanied by a decrease in performance. The relative trends between GRPO and GRPOMask are similar for DMATH, but differ in absolute terms. We include further discussion in App. F, where we also demonstrate that MULT (which does not exhibit asymmetries) benefits far less from negative gradients.

2) Negative gradients promote diverse responses during RL training, encouraging exploration at two levels: (i) within a rollout; and (ii) across rollouts. For (i), we observe that removing the negative gradient results in an entropy collapse over the next-token distribution (Fig. 3 (d)). This leads to responses with a repeating stream of tokens when extrapolating the trained model to larger budgets (see App. F). For (ii), we measure the cumulative unique attempts on the CDOWN test set as we train the model (App. F) and find more unique attempts when training with negative gradients.

3) LLMs trained with negative gradients extrapolate better. The bridge between exploration and meta-exploration lies in the use of asymmetries present in the base model. Exploration afforded by negative gradients, in the presence of asymmetries like the VG gap, incentivizes meta-exploration, because longer responses with more chained asymmetries (verification-generation steps) discover correct solutions and get positively rewarded. Recall from Sec. 1 that if a model has learned to explore in-context (meta-exploration), it should benefit from additional test compute since under large VG gaps are present. We confirm this in Fig. 3(a) (dotted lines), where we see that when testing on hard test problems in DMATH, on a budget that is $2\times B_{\text{tr}}$, the performance gap widens with negative gradients, in comparison to the masked version. Refer to App. C for an analysis of negative gradient dynamics in a didactic setting.

4. Coupled Curriculum Training Structures Exploration in Long Length RL

In the presence of asymmetries, training with negative gradients produces models that can extrapolate beyond their training budget. However, just negative gradients are not enough: as we see in Fig. 4(a), different training budgets B_{tr} lead to different levels of performance on B_{tr}, as well as extrapolated test compute. *So how should we set the budget B_{tr} to attain strong extrapolation performance?* And in correspondence with token budgets, *what prompts should we be training on for a given budget?*



(a) (b)

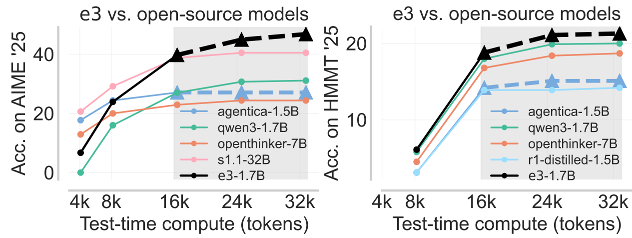
Figure 4: ***RL training on different data and length budgets.*** **(a):** Optimal results come from balancing optimization (better at shorter budgets) and in-context exploration (better at longer budgets). **(b):** Training on hard problems at the 8k budget kills in-context exploration. Refer to App. G for the length distributions.

Setup. We evaluate on DMATH, CDOWN, with different training budgets and data compositions. We split DMATHevenly across three levels of hardness by Qwen-R1-Distilled-32B accuracy. For CDOWN, we judge problem difficulty based on the number of terms in the equation. We use the GRPO (Shao et al., 2024) algorithm to train models on all compute budgets and datasets (see App. G for details).

Training solely at low or high values of B_{tr} is not desirable. We first run RL training for 300 iterations on the easy DMATH problems at different training budgets B_{tr}= 4k, 8k,16k (see Fig. 4(a)). While training at the short budget B_{tr}= 4k attains the best performance at the same test budget of 4k tokens, it "kills" exploration and leads to poor extrapolation performance (no gains from 8k to 40k). On the other extreme, training at B_{tr}= 16k introduces significant optimization challenges, typical of long horizons policy gradients suffering from high gradient variance (Agarwal et al., 2021). We find that B_{tr}= 8k attains the best scaling when extrapolating test compute, implying that we need to strike a balance between the length budget available for negative gradient to encourage chaining asymmetries (infeasible in $\leq$4k tokens) and mitigating optimization challenges.

Training naïvely on a static data mixture is insufficient. Having identified a reasonable B_{tr} of 8k, we now turn to studying the effect of data compositions. We compare the naïve training data mixture with all difficulties (easy+med+hard) against easy, easy+med at B_{tr}= 8k. Matching train and test composition is ideal for better "in-distribution" performance, *i.e.*, when evaluating models at B_{tr} (see App. G). Surprisingly, the same is not true for extrapolation on out-of-distribution (OOD) problems at larger test-time budgets. As shown in 4(b), the model trained on *only easy* problems obtains the best performance on OOD AIME'25 when extrapolating compute to 40k.

How can we avoid challenges with training on a fixed dataset and length budget? One approach is to incorporate a budget curriculum that varies B_{tr} over training. However, this alone is insufficient because, as shown above, training on hard problems with short budgets suppresses length and in-context exploration. On the other hand, we can design



(a) (b)

Figure 5: ***RL training with coupled curricula.*** In the above figure, the shaded area indicates the extrapolation regime. `e3` achieves *state-of-the-art* performance across models < 2B. on **(a)** AIME '25 and **(b)** HMMT '25

a curriculum over the difficulty level and keep the training budget fixed at a high enough value. However, this leads to learning over-exploratory traces tailored to easy problems (see App. G for a detailed study of this on CDOWN). We also show an experiment comparing our proposed fix below with only budget or data curricula in Fig. 5(d).

e3: coupled curriculum for test-time extrapolation. Motivated by our findings above, we propose a ***coupled curriculum*** which varies B_{tr} and problem difficulty in a coordinated fashion as training progresses. We refer to our entire recipe as `e3`: *exploration enables extrapolation*. This encompasses asymmetries, negative gradients, and a prescription for the coupled curriculum. Our recipe `e3` fine-tunes the base model on easy problems in DMATH at a budget of 8k, and subsequently continues training on medium and hard problems in DMATH with a budget of 16k. Refer to App. G for the theoretical motivation behind this curriculum.

Final results with e3. In Fig. 5(a,b), we compare the performance of Qwen3-1.7B fine-tuned using `e3` with open-source models, including 7B and 32B models. As shown, `e3` achieves state-of-the-art performance on AIME' 25 and HMMT' 25, within a model class of size <2B. We outperform the best model in this class by >10% on AIME '25 in terms of peak performance, and show that our model, trained only up to a budget of 16k, extrapolates better than other models including s1.1-32B (Muennighoff et al., 2025) and OpenThinker-7B (Team, 2025) when we extend the compute budget up to 32k. Refer to App. G for more details. Finally, Fig. 6 shows that compared to budget forcing, which is a prompting technique introduced in s1 (Muennighoff et al., 2025) to enable extrapolation, `e3` achieves significantly better scaling, even without applying budget forcing to it. Refer to App. L for a discussion on the conclusion and limitations.

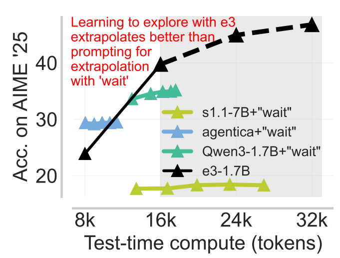


Figure 6: `e3` (w/o "wait") is superior when extrapolating to larger budgets, compared to budget forcing with "wait" prompt 2/4/6/8 times.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

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

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Amin, S., Gomrokchi, M., Satija, H., Van Hoof, H., and Precup, D. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

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

Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., and Goodman, N. D. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

Gao, Y., Meyer, C. M., Mesgar, M., and Gurevych, I. Reward learning for efficient reinforcement learning in extractive document summarisation. *arXiv preprint arXiv:1907.12894*, 2019.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. *CoRR*, abs/1802.07245, 2018.

Kim, S., Wu, I., Lee, J., Yue, X., Lee, S., Moon, M., Gashteovski, K., Lawrence, C., Hockenmaier, J., Neubig, G., et al. Scaling evaluation-time compute with reasoning models as process evaluators. *arXiv preprint arXiv:2503.19877*, 2025.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Li, Q., Zhai, Y., Ma, Y., and Levine, S. Understanding the complexity gains of single-task rl with a curriculum. *arXiv preprint arXiv:2212.12809*, 2022.

Liu, Z., Chen, Y., Shoeybi, M., Catanzaro, B., and Ping, W. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint*, 2024.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Luo, M., Tan, S., Huang, R., Patel, A., Ariyak, A., Wu, Q., Shi, X., Xin, R., Cai, C., Weber, M., Zhang, C., Li, L. E., Popa, R. A., and Stoica, I. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mi 2025a. Notion Blog.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W. Y., Roongta, M., Cai, C., Luo, J., Zhang, T., Li, L. E., Popa, R. A., and Stoica, I. DeepScaleR: Surpassing O1-Preview with a 1.5B Model by Scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025b. Notion Blog.

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, 2025. URL https://arxiv.org/abs/2501.19393.

OpenAI, :, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Zhang, Y., Wang, Y., Shao, Z., and Li, Z. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.

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.

Ren, Y. and Sutherland, D. J. Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Setlur, A., Rajaraman, N., Levine, S., and Kumar, A. Scaling test-time compute without verification or rl is suboptimal. *arXiv preprint arXiv:2502.12118*, 2025.

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.

Shi, T., Wu, Y., Song, L., Zhou, T., and Zhao, J. Efficient reinforcement finetuning via adaptive curriculum learning, 2025. URL https://arxiv.org/abs/2504.05520.

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.

Song, Y., Zhang, H., Eisenach, C., Kakade, S., Foster, D., and Ghai, U. Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*, 2024.

Swamy, G., Dann, C., Kidambi, R., Wu, Z. S., and Agarwal, A. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024.

Swamy, G., Choudhury, S., Sun, W., Wu, Z. S., and Bagnell, J. A. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.

Tajwar, F., Singh, A., Sharma, A., Rafailov, R., Schneider, J., Xie, T., Ermon, S., Finn, C., and Kumar, A. Preference Fine-Tuning of LLMs Should Leverage Suboptimal, On-Policy Data, ICML 2024.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Wang, Z., Yang, Z., Huang, Z., Huang, Z., Xu, Z., and Yang, Z. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Team, O. Open Thoughts. https://open-thoughts.ai, February 2025.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025.

Xie, T., Gao, Z., Ren, Q., Luo, H., Hong, Y., Dai, B., Zhou, J., Qiu, K., Wu, Z., and Luo, C. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Yeo, E., Tong, Y., Niu, M., Neubig, G., and Yue, X. Demystifying long chain-of-thought reasoning in llms, 2025. URL https://arxiv.org/abs/2502.03373.

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.

Yuan, Z., Yuan, H., Li, C., Dong, G., Tan, C., and Zhou, C. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025a.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025b. URL https://arxiv.org/abs/2503.18892.

Appendices

A. Related Work

Scaling test-time compute via Long CoT reasoning. Prior work explores a number of avenues for scaling test-time compute, including majority voting (Wang et al., 2022), best-of-n sampling, and beam search (Setlur et al., 2024; Snell et al., 2024), as well as sequential self-correction (Qu et al., 2024; Kumar et al., 2024). More recent results indicate that training models to use test-time compute to generate longer chains of thought (CoT) that combine verification, search, and self-correction – all in a free-form manner, performs better (DeepSeek-AI et al., 2025; Team et al., 2025; OpenAI et al., 2024), resulting in widespread open-source reproduction efforts (Face, 2025; Yeo et al., 2025; Zeng et al., 2025b; Luo et al., 2025b). We situate our work in the paradigm of long CoT reasoning and study the role of algorithms (RL or SFT), data composition, and design of the training procedure.

Test-time extrapolation. The true benefit of test-time scaling is consistently improving performance as we extrapolate test compute. While prior work tests the model's performance on budgets longer than the training budget (Zeng et al., 2025a; Luo et al., 2025a), they do not explain the relationship between the training recipe and the extrapolation capabilities. In our work, we provide a clear recipe and explain the mechanism behind why our recipe enables test time extrapolation. Other works perform extrapolation by explicitly prompting models to generate more tokens when a response terminates (Muennighoff et al., 2025; Aggarwal & Welleck, 2025). In this work, we show that models that learn to explore in-context extrapolate test compute better than prompting-based approaches. In particular, we study the role played by the base model, training algorithm (RL), as well as data mixtures and token budgets, on the ability to extrapolate. Furthermore, prior work (Setlur et al., 2025) has investigated how performance scales with budgets when train and test budgets are matched, which is different from the OOD setting this work where test budgets are significantly longer.

Exploration in test-time scaling. Long CoTs allow models to explore various reasoning paths before exploiting and committing to a final answer. While prior works have shown the importance of the base model's ability to conduct exploration (Gandhi et al., 2025; Liu et al., 2025), we discover the crucial enabling factor is the presence of *asymmetries* in the model. Next, we show that the negative gradient in RL incentivizes the model to chain together multiple asymmetries, which in turn leads to an increase in the length of the response. In contrast, SFT alone does not provide this kind of chaining or exploration benefits. Our analysis is orthogonal to theoretical works Setlur et al. (2025); Swamy et al. (2024), which shows that RL performs better than SFT, but from a statistical perspective, whereas our argument is more focused on the learning dynamics. Concurrent work builds techniques to boost exploration during RL via advantage normalization (Li et al., 2022; Yu et al., 2025) or PPO clipping (Yu et al., 2025), and these techniques can be combined with `e3`, but they do not highlight the role of negative gradients in learning to explore. Finally, Wang et al. (2025) briefly remarks about the role of policy gradient loss and entropy when running RL with only a few examples. Our study formally investigates the underlying mechanism of negative gradients increasing length and entropy with controlled experiments and thoeretical results.

Data and length curricula. Recent works have also investigated using a curriculum on problem difficulty (Team et al., 2025; Xie et al., 2025; Shi et al., 2025) and context window length (Luo et al., 2025b; Liu et al., 2024) during RL training. Their motivation stems primarily from an efficiency standpoint: avoiding zero advantage updates (Shi et al., 2025; Yu et al., 2025), efficient optimization (Luo et al., 2025b), or efficiency of using test-time compute (Qu et al., 2025). While we do make similar observations regarding each curriculum individually, perhaps our most interesting finding is that carefully coupling both data and budget curricula can lead to much better performance and extrapolation, beyond merely some gains in compute efficiency. We show that training on hard problems with short budgets often yields terse solutions that fail to extrapolate, while easy problems with long budgets can cause optimization issues or verbose outputs. Thus, curricula must be carefully designed to support effective extrapolation. Conceptually, our curricula are most related to dense progress rewards (Qu et al., 2025; Setlur et al., 2024), in the sense that curricula incentivize different degrees of progress for different questions, at different points in training. We believe this is a good avenue for future work to pursue.

B. Optimizing & Extrapolating Test-Time Compute

Post-training scaling test-time compute. RL and SFT are categories of post-training algorithms that refine a pre-trained base LLM π_b into a reasoning model, in particular one that utilizes more test-time compute by producing long chains-of-thought to succeed. Typical outcome-reward RL trains LLM π (initialized with π_b) to maximize performance on outcome 0/1 reward $r^\star(\mathbf{x}, \mathbf{y})$, for inputs $\mathbf{x} \sim \rho$ and response $\mathbf{y} \sim \pi(\mathbf{y} \mid \mathbf{x})$ restricted to an apriori fixed maximum token length or *training budget* B_{tr} (Yu et al., 2025; Luo et al., 2025b). On the other hand, SFT fine-tunes π_b on long thinking traces from more capable models or humans to distill their reasoning capabilities (Team, 2025; Muennighoff et al., 2025), where the maximum length of the expert traces also implicitly induces a training budget B_{tr}, similar to RL. **Our goal**, is to train models that can improve performance when we extrapolate test-compute beyond B_{tr}. Even though the true promise of test-time compute is extrapolation performance, we find that ***current thinking models fall short on extrapolation.*** We evaluate multiple models on a test budget of 32K, ≈1.5-2×B_{tr} across all models. We plot our results on AIME25 in Fig. 7 (see App. D for a detailed comparison) and note that most of the performance gains lie in the training budget, and the gains are minuscule as we test beyond that.

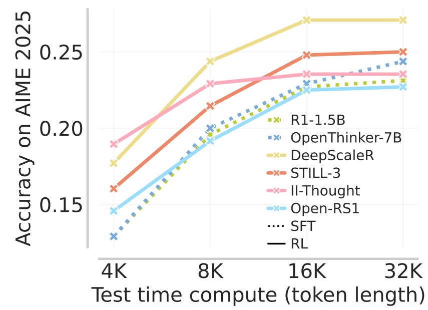


Figure 7: Accuracy on AIME 2025 of various open-source models at different test time compute budgets. Performance gains diminish as the test-time budget increases, with virtually no gains from 16k to 32k.

Negative gradient in RL. A key distinction between SFT and RL is the ***negative gradient***, which corresponds to the part of the policy gradient coming from traces that fail. In Eq. 1 we present a generalized version of the policy gradient adopted by most RL post-training methods: REINFORCE (Ahmadian et al., 2024), PPO (Schulman et al., 2017), and GRPO (Shao et al., 2024). From this we note that on a prompt $\mathbf{x}$, RL training observes two types of gradients: (i) the positive gradient which maximizes the likelihood of a correct responses $\mathbf{y}$ with a positive advantage $A(\mathbf{x}, \mathbf{y})$, and (ii) the negative gradient which *pushes down* the likelihood of an incorrect response with a negative advantage $A(\mathbf{x}, \mathbf{y})$. Here, $\mathbf{y}$ can be sampled *on-policy* $\pi = \tilde{\pi}$ or *off-policy* $\pi \neq \tilde{\pi}$. Thus, we can view SFT as a purely positive gradient method that only maximizes likelihood on correct reasoning traces. In Sec. 3, we show why the negative gradient is largely responsible for driving up response lengths and in-context exploration during RL, thereby enabling RL-trained models to explore more at test-time and extrapolate better compared to SFT-based ones.

$$\mathbb{E}_{\mathbf{y} \sim \tilde{\pi}(\cdot \mid \mathbf{x})} \left[A_i(\mathbf{x}, \mathbf{y}) \cdot \nabla_\pi \log \pi(\mathbf{y} \mid \mathbf{x}) \right] \quad \text{(general form of policy gradient in RL)} \tag{1}$$

C. Analyzing Negative Gradient Dynamics in the p^k Model

We give an informal example of how an LLM can leverage VG gap to improve performance through longer in-context exploration: the ***p^k-model***. We view the LLM as sequentially guessing candidate responses $a_1, a_2, \ldots$, each with failure probability p, up to at most terminal k responses. We assume that this model admits perfect verification (***perfect VG gap***), which means that the learner can correctly assess whether each subsequent sequential response is correct and decide when to stop accordingly. In a simplified setting where attempts are independent, failure probability (= p^k) decays exponentially as k increases, as on CDOWN. In contrast, if verification is difficult (*i.e.*, no VG gap), increasing k provides little benefit, since the model cannot adjudicate whether one guess is any better than another. In extreme scenarios, the only way to improve performance is by lowering p (better first guesses as seen on MULT).

Analyzing Negative Gradient Dynamics in the p^k Model. We introduce a didactic setup where verification is perfect but attempts are not independent, akin to LLMs we train in practice. We consider a Markov decision process (MDP) with action space $\bar{\mathcal{A}} = \mathcal{A} \cup \{\texttt{stop}\}$, where $\mathcal{A} = [100]$ are standard actions and stop is an early "stopping" action (like EOS) that terminates the trace. For simplicity, we consider policies parametrized as a softmax bigram model $\pi_M(a_{t+1} \mid a_t)$, with

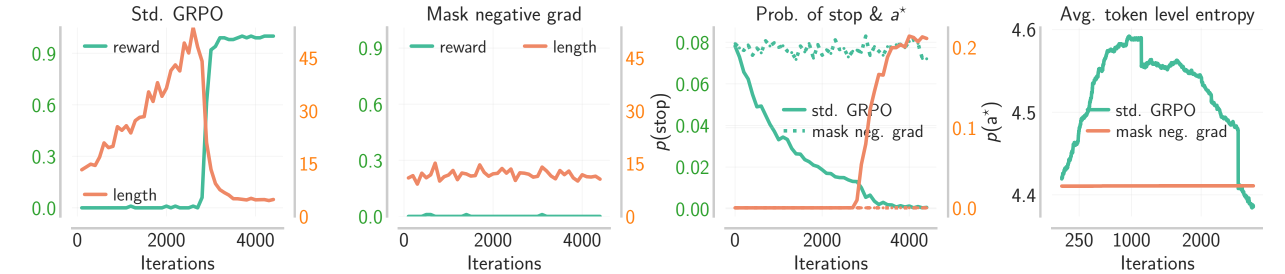


Figure 8: ***Negative gradients in a bi-gram model.*** Negative gradients push down $p(\texttt{stop})$ during training (c), increasing length (a) and entropy of the next action distribution (d) to accommodate more in-context exploration, only decreasing them when $a^\star$ is discovered. In contrast, positive gradients rarely change $p(\texttt{stop})$ or entropy.

details deferred to the App. F. In this bi-gram model, the current state s_t always matches the previous action a_{t-1}, and $a^\star \in \mathcal{A}$ denotes the optimal action. In a rollout $a_1, ..., a_t$, the initial action a_1 is sampled from a fixed π_0. For $t>1$, a learner policy samples an action $a_t \sim \pi(\cdot|a_{1:t-1}) \in \triangle(\mathcal{A})$. The MDP terminates with reward 1 at time t if $a_t = a^\star$, and with reward 0 if $a_t = \texttt{stop}$ (stops too early), or $t > B_{\text{tr}}$ (budget is exhausted before a correct response). The model **learns to explore** if it learns to never play $\texttt{stop}$ for any t (no early stopping), until $a^\star$ is observed, *i.e.*, increasing k in p^k. **Refining the guess** amounts to upweighting $\pi(a^\star \mid a_{1:t-1})$ without reducing $p(\texttt{stop})$, *i.e.*, improving p in p^k.

1) Negative gradient increases length until $p(a^\star)$ **is reasonably high.** In Fig. 8(a) standard GRPO (B_{tr}=100)increases average response length from 15 to 45 at budget, driven by the drop in the marginal probability of stopping early $p(\texttt{stop})$ (Fig. 8(c)). After multiple RL iterations with negative gradients, the average number of attempts per trace is sufficiently large, and the learner can sample $a^\star$ with non-trivial probability in any given trace. Once this happens (Fig. 8(c)), in our simple bigram setup, the model rapidly upweights the likelihood of one-step transitions to $a^\star$, resulting in a phase transition where reward increases as length drops. In our LLM benchmarks, however, we do not see the same phase transition since finding "shortcuts" to correct responses is considerably more difficult. In contrast, GRPOMask (Fig 8(b)) fails to improve reward or increase length.

2) Negative gradient improves coverage by increasing entropy of $\pi(\cdot \mid a_{1:t-1})$**.** When π_M samples a highly likely yet incorrect action, the negative gradient computed on this sample increases entropy by moving probability mass onto less-seen modes of the distribution, including $a^\star$. We show this formally in Theorem C.1 where we prove that upon sampling a highly likely incorrect action with probability p, GRPO update with a negative gradient results in an entropy increase of $\approx p^2$ when all other actions, including $a^\star$ are highly unlikely. We note this empirically as well in Fig. 8(d), where conditional entropy increases across states, until $a^\star$ is discovered, after which it drops sharply as the positive gradient rapidly moves mass onto $a^\star$ within a few iterations.

Theorem C.1 (Negative gradient increases entropy when $a^\star$ is unlikely; formal version of Thm. H.3)**.** *At state* $\mathbf{s}$*, if the most likely action under* π *is* $a_1 =: \arg\max_{a'} \pi(a'|\mathbf{s}) \neq a^\star$*, then, for any* π*, a negative stochastic gradient step increases the entropy of* $\pi(\cdot|\mathbf{s})$ *with prob.* $\geq \pi(a_1|\mathbf{s})$*. Additionally, in a suitable regime of* π*, the increase* $\gtrsim (\pi(a_1|\mathbf{s}) - \pi(a_2|\mathbf{s}))^2$*, where* a_2 *is second most likely after* a_1*. In contrast, in the absence of the negative gradient, the entropy is preserved with prob.* $1 - \pi(a^\star|\mathbf{s})$.

D. Testing Extrapolation of Open-Source Models

Extrapolation on AIME 2025 Extrapolation (i.e. the chaining of generation, verification, refinement, etc.) can potentially extend LLM performance after training, and do so beyond the context length the model was originally trained on. To evaluate this properly, we need sufficiently challenging problems that allow meaningful expressiveness in reasoning beyond small context lengths. The math problems associated with AIME align with this, and our evaluations prioritize AIME 2025 to attempt to mitigate any potential data contamination in the models' training sets from previous years of AIME. The goal of the experiment is to measure the extent to which test-time compute influences overall model performance as context length increases, with the expectation that increasing output length allows models to "reason" for longer periods, continuing the extrapolation process, and ultimately arriving at the correct answer more frequently.

Experiment Setup Inference for every open-source model was performed using Oumi through data-parallel SGLang. All models had inference run with a max output length of approximately 32k tokens, though some are slightly lower due to this exceeding their max context length when combined with the prompt. The exact inference hyperparameters are described in

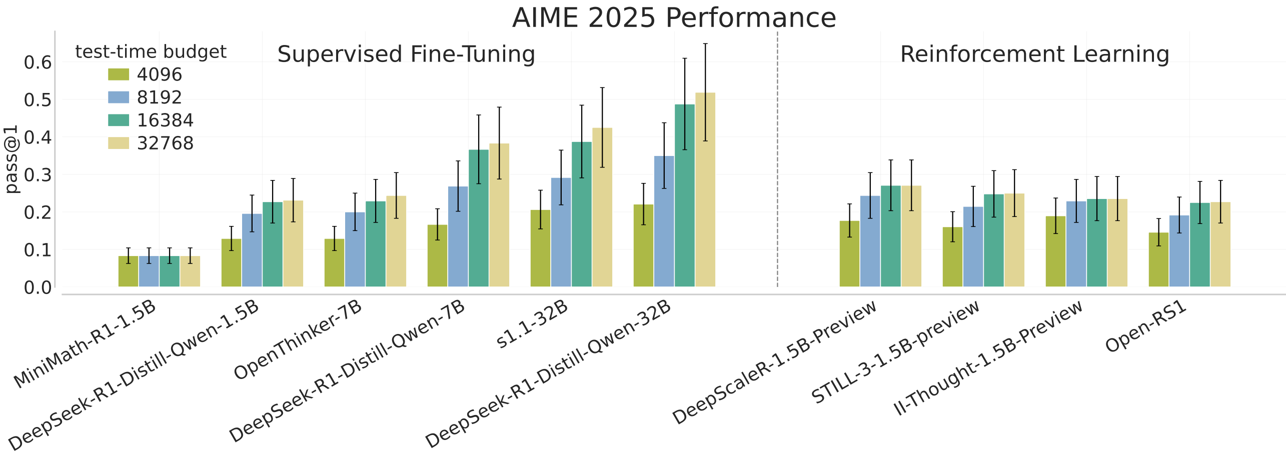


Figure 9: Performance (pass@1) on AIME 2025 at different test-time compute budgets across multiple open-source models of different sizes, trained with SFT or RL.

Table 1. After inference, the model responses were truncated from the right side until the number of remaining tokens present was equal to the specified test-time budget. 16 responses were collected for every problem in AIME with the specified inference settings, and the Pass@1 rate was calculated by averaging over these 16 responses. Final answers were extracted using a regular expression for the boxed portion of the answer, with correct answers marked as passing and incorrect or incorrectly parsed answers marked as nonpassing. The prompt used is in Box D.1, and the problems were taken from the FVU AIME 2025 dataset on HuggingFace[1].

Box D.1: AIME Evaluation Prompt Template

You will be given a math problem. Solve the problem step by step. Output your final answer in the form of \\boxed{your answer}. Problem: {problem}

Model	Temp.	Top p	Rollouts	Max New Tokens	Model Max Length
MiniMath R1-1.5B	0.6	0.95	16	32768	40960
DeepSeek R1-Distill-Qwen-1.5B	0.6	0.95	16	32768	40960
OpenThinker-7B	0.6	0.95	16	31000	32768
DeepSeek-R1-Distill-Qwen-7B	0.6	0.95	16	32768	40960
s1.1-32B	0.6	0.95	16	31000	32768
DeepSeek-R1-Distill-Qwen-32B	0.6	0.95	16	32768	40960
DeepScaleR-1.5B-Preview	0.6	0.95	16	32768	40960
STILL-3-1.5B-preview	0.6	0.95	16	32768	40960
II-Thought-1.5B-Preview	0.6	0.95	16	32768	40960
Open-RS1	0.6	0.95	16	32768	40960

Table 1: Inference parameters used for generating the extrapolation plots in Figure 7.

Results The results in Figure 9 show that as the maximum number of output tokens increases, every model capable of "reasoning" is able to attain a higher Pass@1 rate, with performance generally saturating at 16k tokens with relatively minor improvements at 32k. We do not observe this with MiniMath-R1-1.5B, and we suspect this is due to its fine-tuning focusing solely on smaller math problems trained with supervised fine-tuning, likely resulting in catastrophic forgetting of the ability to continuously extrapolate. Interestingly, we do not see a strong improvement in extrapolation behavior among models tuned with reinforcement learning compared to DeepSeek R1-Distill-Qwen-1.5B, which was trained with supervised fine-tuning. We suspect that this is likely due to the nature of the distillation data from the R1 model, which, if varied

[1] https://huggingface.co/datasets/FVU/AIME_2025

sufficiently in length, could avoid the length bias normally learned from supervised fine-tuning, while still teaching the model to perform extrapolation.

E. Additional Experiments and Details for Section 2 (Chained Asymmetries)

E.1. Details on MULT and MULT-V

Data collection. Both MULT and MULT-V consist of multiplication traces for solving a 5-digit $\times$ 5-digit multiplication problem. For the MULT task, we use a Llama3.2-3B instruction tuned model where the number of intermediate verification attempts is much lower in a trace when asked to solve a multiplication problem. In fact, it is not hard to see that, in general, for multiplication, generation of a trace may be as hard as verifying a generated one, as the only way to verify the entire trace is to re-attempt the multiplication or carry out a division with the computed target. We contrast this task with the MULT-V task, where the Llama3.2-3B models are first finetuned on traces from Qwen-32B-R1-Distilled and GPT-4o models. These traces contain multiple verification attempts that verify intermediate steps solving smaller multiplication problems, and the steps are part of an entire trace that attempts to solve the main multiplication problem involving two 5-digit numbers. For collecting data we used the prompt in Box E.1. In App. K Example 2, we also provide an example multiplication trace with verification attempts sampled by the base model in MULT-V. As we will see in Fig. 14, the absence of asymmetries in MULT leads to lower accuracy and verifications when compared to MULT-V, where asymmetries are present.

Box E.1: Prompt for generating MULT-V data

Multiply {num1} and {num2}. Please reason step by step, and put your final answer within \\boxed{}. At each step, try to verify your response if possible and prefix the line with "Check:". <think>

Hyperparameter	Values
train_batch_size	256
ppo_mini_batch_size	64
learning_rate	5.0e-6
kl_loss_coef	0.001
entropy_coeff	0.001
temperature	1.0
rollout.n	16

Table 2: Verl (Sheng et al., 2024) hyperparameters used for MULT and MULT-V.

Training details. Hyperparameters for our experiments on MULT and MULT-V are given in Table 2.

E.2. Details on CDOWN

Training details. Hyperparameters in CDOWN experiments follow the table below unless otherwise specified. In all of our CDOWN experiments, we take the fine-tuned Llama3.2-3B base model from (Gandhi et al., 2025). For Fig. 2, we trained with $B_{\mathrm{tr}} = 512, 1024, 2048$ on problems with $3, 4, 5, 6$ candidates. The total number of datapoints we used was 40000, which were evenly split across the four difficulties.

Evolution of chained asymmetries at test time. To measure the benefits of chained asymmetries on CDOWN, we plot the pass@k accuracy of the base LLM, shown in Figure 10, and observe that performance increases with the chained asymmetries budget. Moreover, as training progresses, responses with more chained asymmetries enjoy a greater improvement. If we move across any diagonal parallel to the main diagonal from top left to bottom right, we move across a constant attempt budget (*e.g.,* moving from 16 chained asymmetries $\times$ 1 pass to 8 chained asymmetries $\times$ 2 passes). Having sequential chained asymmetries become increasingly better than parallel rollouts as training progresses, indicating the exploitation of asymmetries in RL training. See example of chained asymmetry in App. K, Example 1.

Hyperparameter	Values
train_batch_size	128
ppo_mini_batch_size	32
learning_rate	1.0e-6
kl_loss_coef	0.001
entropy_coeff	0
temperature	0.6
rollout.n	8

Table 3: Verl (Sheng et al., 2024) hyperparameters used for CDOWN.

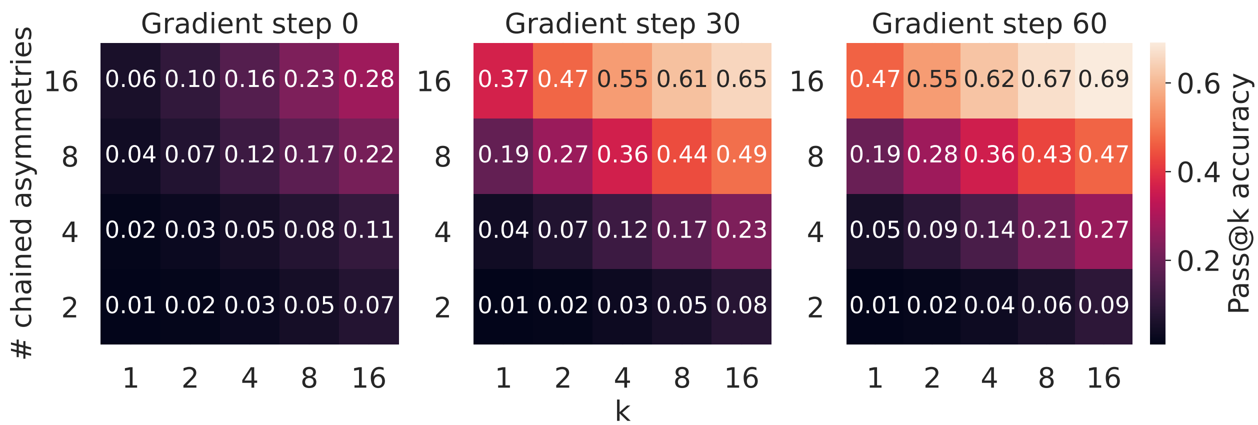


Figure 10: **Evolution of asymmetries during training on CDOWN:** More chained asymmetries lead to a greater improvement in pass@k performance across gradient steps.

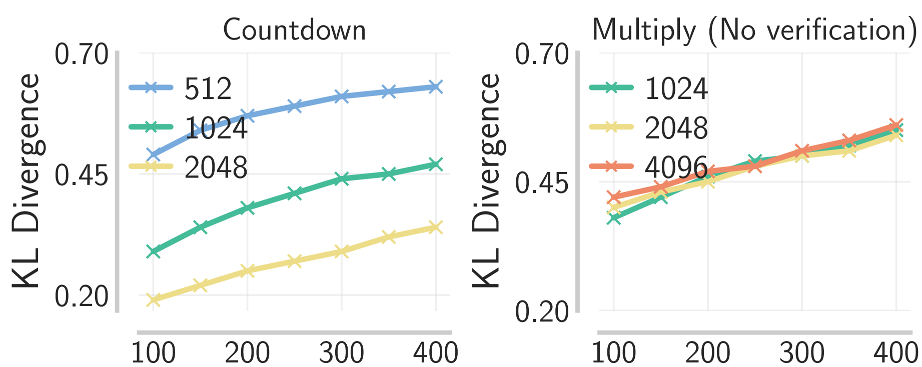


Figure 11: **KL-divergence with base LLM on CDOWN and MULT:** When running RL training on CDOWN and MULT with multiple training budgets (512, 1024, 2048 on CDOWN and 1024, 2048, 4096 on MULT) we note that the KL divergence

E.3. In the Presence of Asymmetries, KL Divergence with Base LLM Reduces as Training Budget Increases

In Fig. 11, we also interestingly observe that training with higher B_{tr} results in a smaller token KL-divergence from π_b all throughout training on countdown. On multiplication in the absence of asymmetries, the KL-divergence values are roughly similar for all B_{tr}. This means that when the verification-generation asymmetry is present, the training process deviates less from π_b at each token, but is able to "chain" multiple verification and generation attempts together to improve accuracy, by learning to explore over the space of basic skills. Prior work argues that a model that deviates less from the base pre-trained model generalizes better on unseen prompts (Gao et al., 2019). If we were to apply this argument in our case, this means that models that are able to use asymmetries better should result in better performance on unseen prompts, especially when operating at higher test compute.

F. Additional Experiments and Details for Section 3 (Negative Gradient)

F.1. Details for CDOWN

We trained models for 90 steps on problems with 5 candidate numbers with a training budget of 2k.

Cumulative unique attempts plot. Fig. 12 (left) was filtered on incorrect traces on problems with < 50% success across gradient steps. We select only incorrect traces to capture the ability of the model to explore for the correct trace, rather than to output diverse correct traces once one is found. We filter for problems with < 50% success across training for GRPO and GRPOMask because otherwise the algorithm with better rewards would see more problems with lower cumulative unique attempts, as the correct traces are discovered early and subsequently reinforced.

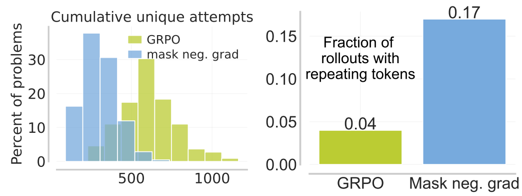
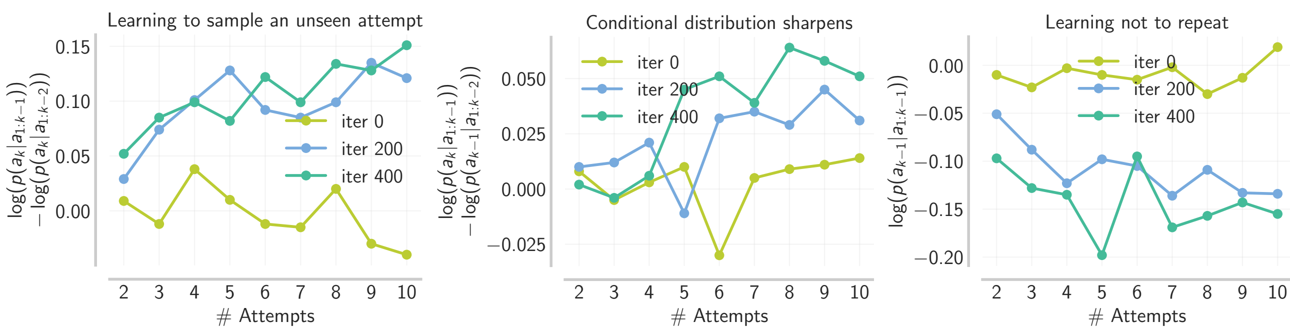


Figure 12: Negative gradients encourage distinct responses: they increase the cumulative number of unique attempts on CDOWN (left) and reduce responses that end with a repeating stream of tokens on DMATH (right).

Conditional distribution given past attempts. We run ablations on the conditional distribution of a new attempt given past attempts in three different settings, shown in Fig. 13. In (a), we plot $\log p(a_k|a_{1:k-1}) - \log p(a_k|a_{1:k-2})$, which should average to roughly 0 if the attempts are independent. As training progresses, this quantity grows, indicating a correlation between attempts, especially with larger k (potentially because the new attempt can attend to more previous attempts, and thus becomes more dependent on them). In (b), we plot $\log p(a_k|a_{1:k-1}) - \log p(a_{k-1}|a_{1:k-2})$, which also grows over time. This indicates that the conditional distribution $p(\text{new attempt}|\text{past attempts})$ as the number of past attempts grows, aligning with the higher dependency on past attempts at larger k in (a). In (c), we plot $\log p(a_{k-1}|a_{1:k-1})$ to check whether the model would repeat its latest attempt. We observe that as training progresses, the model learns not to repeat itself.



(a) (b) (c)

Figure 13: ***Probing the conditional distributions conditioned on past attempts in*** **CDOWN. (a):** New attempts are not independent of past attempts **(b):** Model becomes more certain of what to try next given more past attempts **(c):** Model learns not to repeat past attempts

F.2. Additional Experiments with MULT

In Section 3 we saw that training with the negative gradient leads to more exploration during RL training, which in turn leads to the amplification of any chained asymmetries that may be present in the base model, *e.g.*, more generation-verification steps. In particular, we noted the increase in the number of verification steps in Fig. 3(b). To see how negative gradients affect the dynamics of response length and number of chained asymmetries in the absence of a strong VG gap, we compare running GRPO with and without negative gradients on our multiplication task MULT where the VG gap is weaker in the base model.

We plot results in Fig. 14, where we note two trends when running RL training with and without negative gradients on MULT(without VG gap), and MULT-V(with VG gap) using a training budget of 4096 tokens. First, we note that the number of verifications is higher when we use negative gradients in a setting with a large VG gap. When the VG gap is absent, the number of chained asymmetries (verification-generation steps) are roughly the same with and without masking the negative gradient. Second, we note that the accuracy is much higher with negative gradients in the presence of VG gap (MULT-V), and comparable to a run where we mask the negative gradients in the setting where the VG gap is poor (MULT). Together, this tells us that the boost in exploration driven by negative gradients leads to more chained asymmetries when the base model presents some of them, like a large VG gap.

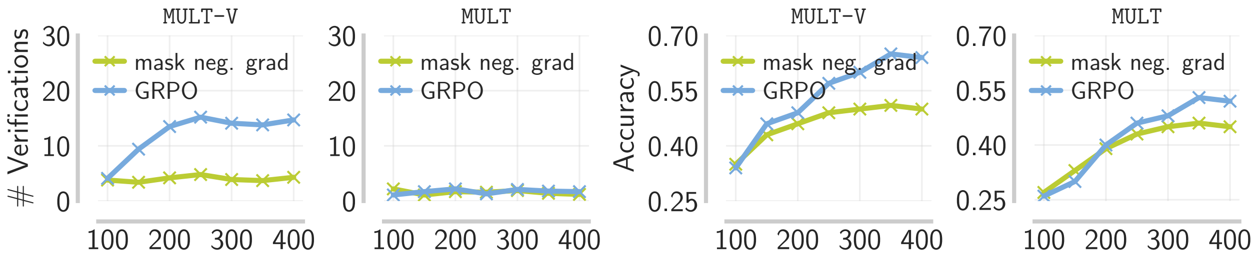


Figure 14: ***Negative gradient amplifies verification when VG gap is large.*** While utilizing the negative gradient amplifies the number of calls to verification in MULT-V, the number of verification calls does not grow over training in MULT. Interestingly, though, we find that when negative gradient is masked out on MULT-V, the number of verification calls is still very low and does not increase, corroborating our findings that exploration driven by negative gradients results in in-context exploration only in the presence of asymmetries in the base model. A similar trend is also observed in terms of the raw accuracy.

F.3. Additional Details for the Didactic Setting in Sec. 3

First, we comment on exploration and meta-exploration in RL, and how negative gradients can connnect one to the other in the presence of asymmetries. Second, we introduce some relevant notations, and provide a high-level proof overview. Finally, we provide the full proof.

Negative gradients boost exploration, which in the presence of asymmetries incentivizes in-context exploration. In Sec. 3 we showed how negative gradients can boost exploration in RL, and in the presence of asymmetries in the base model, lead to more chained asymmetries and longer responses – a phenomenon we call in-context exploration. Here, we present a theoretical result that explains why negative gradient can incentivize the more "traditional exploration" in RL, in our didactic bi-gram model. Since verification is perfect in our bi-gram model, any policy in our policy class always stops at the `stop` token. Thus, an increase in exploration leads to longer traces, and more chained asymmetries. As a result, in this setting, we can view an improvement in exploration as an improvement in meta-exploration (or in-context exploration), driven by negative gradients.

Parameterization of the policy class. We parameterize the policy class as a softmax policy, where the probability of next action a_{t+1}, at state current a_t (in a bi-gram model current state is equivalent to the previous action) is parameterized with the vector of logits $[M(a \mid a_t)]_{a \in \bar{\mathcal{A}}}$, i.e.:

$$\pi_M(a_{t+1} \mid a_t) = \frac{e^{M(a_{t+1}|a_t)}}{\sum_{a' \in \bar{\mathcal{A}}} e^{M(a'|a_t)}}, \quad a_{t+1} \in \bar{\mathcal{A}}, a \in \mathcal{A} \tag{2}$$

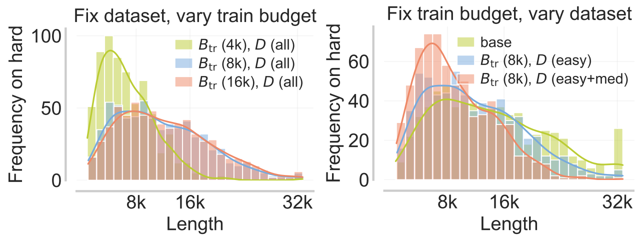
where $M = [M(a^+ \mid a)]_{a^+ \in \bar{\mathcal{A}}, A \in \mathcal{A}}$ can be expressed as a matrix in $\mathbb{R}^{(K+1) \times K}$. Note that the current state can never be the `stop` action, since a `stop` always terminates the MDP.

Training details. We set the initial distribution π_0 to be the uniform distribution over all actions except $a^\star$, i.e., $\pi_0(a^\star) = 0$. For each state $\mathbf{s}$, the policy is initialized with random values of $M(\cdot \mid \mathbf{s})$ in $[-3.0, 3.0]$, and set $M(\texttt{stop} \mid \mathbf{s}) = 4.0$ and $M(a^\star \mid \mathbf{s}) = -4.0$, which mimics the setting where the probability of sampling the stop action is higher than any random action, and the probability of sampling $a^\star$ is lower than any random action. We train with a learning rate of $1e-2$ and use stochastic gradient descent to update the policy where a single update samples a random trajectory τ, starting from a random state sampled from the initial state distribution π_0, until completion and then computes the policy gradient term, by averaging the policy gradient loss over the tokens in the trajectory τ: $1/|\tau| \cdot \sum_{i \in |\tau|} \log \pi_M(a_i \mid a_{1:i-1}) \cdot A(a_i, a_{1:i-1})$.

G. Additional Experiments and Details for Section 4 (Curricula Training)

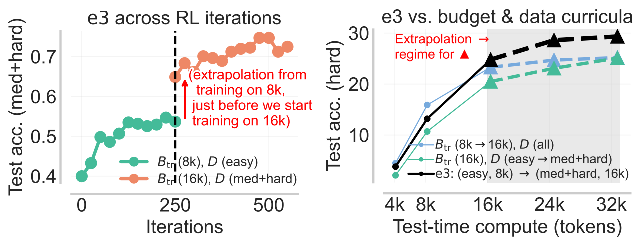
G.1. Response length distributions of different models.

In Fig. 15, we plot the response length distributions of different trained models from section 4 on the hard test set. As shown in (c), training with a low budget can kill exploration on difficult problems, and in (d), training on harder problems can also kill exploration.



(c) (d)

Figure 15: ***Length histograms of RL training on different data and length budgets.*** **(c):** training on small budgets kills exploration under extrapolation **((d):** training on harder problems kills exploration under extrapolation



(a) (b)

Figure 16: ***RL training with coupled curricula.*** In the above figure, the shaded area indicates the extrapolation regime. **(a):** extrapolation gain from switching to a longer budget during training **(b):** coupled curriculum outperforms data and budget curricula

G.2. Additional details on `e3`

Theoretical motivation for coupled curriculum design. We simplify curriculum design by first fixing the dataset at each stage: moving from easy to hard. Now, the key question is to select the appropriate budget for dataset D_i at stage i of the curriculum. Intuitively, we want a budget $B_{\mathrm{tr},i}$ such that training on $B_{\mathrm{tr},i}$ positively rewards in-context exploration, which will in turn improve extrapolation performance and provide a good initialization for the next stage $i+1$. At the same time, to be optimization friendly, we want $B_{\mathrm{tr},i}$ to be minimal while being large enough to accommodate most responses from the given model π_i. Encoding these conditions on $B_{\mathrm{tr}i}$, we propose the following optimization problem.

$$B^{\star}_{\mathrm{tr},i}(D_i) = \arg\min_{B} B \text{ s.t. } J(\pi_i; D_i, 2\cdot B) \geq \kappa \cdot J(\pi_i; D_i, B) \text{ and } B \geq \mathbb{E}_{\mathbf{x}\sim D_i, \tau\sim\pi_i(\cdot|\mathbf{x})}|\tau|, \tag{3}$$

where $J(\pi_b; D, B_{\mathrm{tr}})$ denotes the performance of the base model π_b at budget B_{tr} on dataset D, and $|\tau|$ denotes token count. In practice, we solve the above problem over a fixed set of training budgets: 4k, 8k, 16k, and find this to be a useful heuristic to greedily choose $B_{\mathrm{tr},i}$ in a way that incentivizes in-context exploration at stage i, since it is hard to jointly optimize the budgets across all stages. *E.g.*, setting $\kappa = 1.2$, we find 8k to be the optimal choice for training on easy problems (note that the trained model also satisfies the condition in Eq. 3 at $\kappa = 1.2$, see Fig. 4(a)). Following this, our recipe `e3` fine-tunes the base model on easy problems in DMATH at a training budget of 8k, and subsequently continues training on medium and hard problems in DMATH with a budget of 16k.

In Fig. 16(c), we show that the model already learns to extrapolate at a point during training when we move from the 8k budget to the 16k budget, where there is a >10% performance gain. In (b), we show that a coupled curriculum leads to better (extrapolation) performance compared to solely a length or data curriculum.

G.3. Training Details and In-distribution Performance on Training Budget

We present our hyperparameters for `e3` training runs in Table 4.

Note on in-distribution performance. In Sec. 4 we note that for best extrapolation performance, it is important to vary the mixture of tasks in the dataset, as well as the training budget (max token length) in a a coupled way, over the course of RL training. Here, we note that if we were to only care about in-distribution performance, i.e., performance on a fixed task mixture (of equally proportioned easy, medium, and hard questions in DMATH), then the best way to train is to match the test token budget and prompt mixture with training. In particular, training only on easy problems and a budget of 8k yields a performance of 54.3% on a test dataset consisting of all tasks (from easy, medium and hard splits). But, if we match the test mixture with train, and train on all difficulties, then on the same 8k test budget, we note a performance of 58.9%. Note that the exptrapolation performance (on hard, out-of-distribution AIME '25 questions) of the same models is flipped in Fig. 4.

Hyperparameter	**Values** ($B_{\text{tr}} = 8\text{k}$)	**Values** ($B_{\text{tr}} = 16\text{k}$)
train_batch_size	128	128
ppo_mini_batch_size	32	64
learning_rate	1.0e-6	1.0e-6
kl_loss_coef	0.001	0.001
entropy_coeff	0.002	0.001
temperature	0.6	0.6
rollout.n	8	16

Table 4: Verl (Sheng et al., 2024) hyperparameters used for `e3`.

G.4. Fixed train budget, vary dataset curriculum on CDOWN

In this subsection, we demonstrate that training with a data curriculum based on difficulty with a fixed train budget can lead to over-exploratory output traces, on the example task of CDOWN. With the data curriculum (i.e., fixed budget, vary data), we train first on CDOWN problems with 3 candidate numbers (the "easy" problems) for 60 gradient steps, then those with 6 candidate numbers for 60 gradient steps (the "hard" problems), with a 1k budget across all steps. We compare this with the coupled curriculum in which the first 60 gradient steps are trained with a budget of 256. As shown in Fig. 17, the latter achieves better reward on "hard problems".

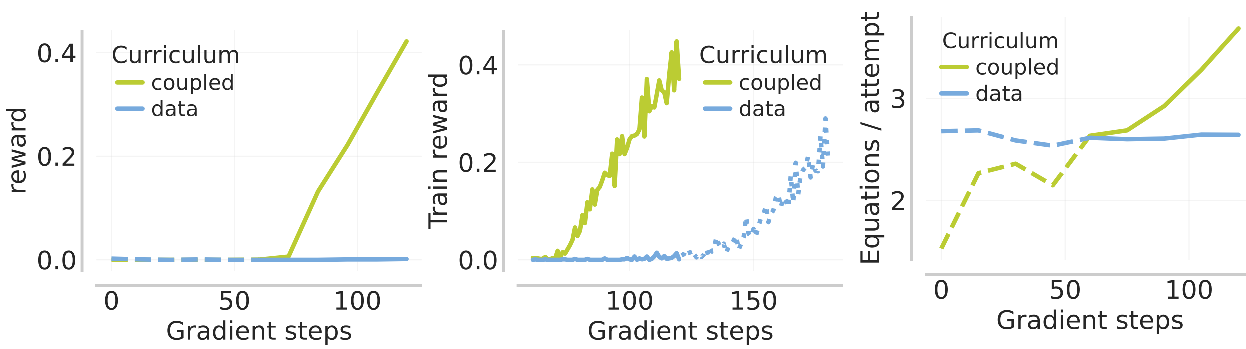


Figure 17: **Coupled vs. data curriculum on CDOWN:** training only on easy problems at large budgets leads to overfitting on "over exploratory" traces, failing to balance explore-exploit tradeoff on harder problems later on. Reward graphs are displayed for hard problems.

Why is data curriculum worse than the coupled curriculum? We can view the learning of correct traces as largely composed of two stages: (i) negative gradients encourage exploration, leading to the discovery of correct traces, (ii) positive gradients reinforce correct traces, once discovered.

For (i), we observe that training on easy problems exacerbates a tendency to perform over-exploratory in-context exploration ("underthinks", see Example 3 in App. K), restricting the discovery of solutions to harder problems. When utilizing a coupled curriculum, this bias propagates to a shorter budget when compared to the data curriculum, since easy problems are trained on 256 rather than 1K tokens. As shown in Figure 17, the average number of equations per attempt (naïvely, with 3 candidate numbers, 2 equations are required to perform a complete attempt vs. 5 equations for 6 candidates) increases noticeably for the coupled curriculum in the second stage, but plateaus for the data curriculum, implying overfitting on "over-exploratory" traces during the first stage.

Furthermore, for (ii), even when nontrivial positive rewards are obtained as we run the data curriculum on hard problems for 60 additional steps (steps 120 to 180), the training reward curve converges more slowly compared to the coupled curriculum (steps 60 to 120), implying that the data curriculum is also worse at reinforcing correct traces if the behavior is over exploratory. While we do not run many controlled experiments to identify why this might be the case, we hypothesize that this is because of imperfect and noisy credit assignment on over-exploratory traces with outcome rewards. It is unclear which segments of the trace should be reinforced vs which segments might simply confuse the model.

H. Omitted Proofs

In this section, we present the formal version of Theorem C.1, and provide a detailed proof for it. First, we introduce some notations and provide a proof overview.

Notations. We use the shorthand $H(M;\mathbf{s})$ to denote the entropy of the conditional distribution over the next action a_{t+1} given the current state $\mathbf{s}$. We also use $M^{(i)}$ to refer to the policy parameters (for the softmax policy in Eq. 2) at iteration i of RL training, and use the shorthand $\pi^{(i)}$ to denote the policy induced by the parameter $M^{(i)}$. We use $\nabla_{M^{(i)}} f(M^{(i)})$ to denote the gradient of function $f(M)$, with respect to M, evaluated at $M = M^{(i)}$. Finally, we use $M_{\mathbf{s}}$ to denote the row of softmax parameters that model the distribution $\pi_M(\cdot \mid \mathbf{s})$, *i.e.*, the row of parameters $M(\cdot \mid \mathbf{s})$ in our parameter matrix M.

Proof overview. Without loss of generality, we fix an arbitrary state $\mathbf{s}$ that is different from `stop`. Given the parameters $M^{(i)}$ at current RL iterate i, we do a Taylor expansion of $H(M^{(i)};\mathbf{s})$ around $M^{(i)}$, and then show that the gradient $\nabla_{M^{(i)}} H(M^{(i)};\mathbf{s})$ is positively correlated with the policy gradient with high probability over the sampling of the action $a \sim \pi_{M^{(i)}}(\cdot \mid \mathbf{s})$, *i.e.*:

$$\langle \nabla_{M_i} H(M^{(i)};\mathbf{s}) \ , \ \nabla_{M^{(i)}} \log \pi(a \mid \mathbf{s}) \ A(\mathbf{s}, a) \rangle \geq 0, \quad \text{whp. over sampling of action } a \sim \pi_{M^{(i)}}(a \mid \mathbf{s}) \tag{4}$$

Before, we prove our result that lower bounds the increase in entropy with negative gradients, we present derivations of the entropy gradient with respect to the model parameters, as well as the policy gradient, which will simplify some calculations in the proof.

Lemma H.1 (Entropy gradient for the softmax bi–gram conditional)**.** *Fix a previous action (because the bi–gram state is* $s_t = a_{t-1}$*, conditioning on the state is equivalent to conditioning on the last action)* $a \in \mathcal{A}$*. Let the (column-wise) logit matrix at time* t *be* $M \in \mathbb{R}^{(K+1)\times K}$*, and define the corresponding softmax conditional distribution*

$$\pi_M(a^+ \mid a) \ = \ \frac{\exp\bigl(M(a^+ \mid a)\bigr)}{Z(a)}, \qquad Z(a) \ = \ \sum_{a' \in \bar{\mathcal{A}}} \exp\bigl(M(a' \mid a)\bigr). \tag{5}$$

Let the Shannon entropy of this conditional distribution be $H\bigl(\pi_M(\cdot \mid a)\bigr)$ *or* $H(M \mid a)$ *Then* $\nabla_M H(M \mid a) \in \mathbb{R}^{K+1}$ *is given by:*

$$\nabla_M H(M \mid a) = = \ -\pi \ \odot \ \bigl(\log \pi + H(\pi)\,\mathbf{1}\bigr) \ = \ -\bigl[\,\pi_i\bigl(\log \pi_i + H(\pi)\bigr)\bigr]_{i \in \bar{\mathcal{A}}}, \tag{6}$$

Proof. Write $p_{a^+} := \pi_M(a^+ \mid a)$ for brevity. By definition of the entropy,

$$H = -\sum_{a^+} p_{a^+} \log p_{a^+}. \tag{7}$$

Insert the softmax expression:

$$\log p_{a^+} = M(a^+ \mid a) - \log Z(a). \tag{8}$$

Hence,

$$H = -\sum_{a^+} p_{a^+} \bigl[M(a^+ \mid a) - \log Z(a)\bigr] \tag{9}$$

$$= -\sum_{a^+} p_{a^+} M(a^+ \mid a) + \log Z(a) \underbrace{\sum_{a^+} p_{a^+}}_{=1}. \tag{10}$$

Rearranging yields the following closed form expression:

$$H = \log Z(a) - \sum_{a^+} p_{a^+} M(a^+ \mid a). \tag{11}$$

Computing the Jacobian of the softmax we get:

$$\frac{\partial \pi_i}{\partial M(j \mid a)} \;=\; \pi_i\big(\delta_{ij} - \pi_j\big), \qquad J := \nabla_{M(\cdot|a)}\pi = \mathrm{diag}(\pi) - \pi\pi^{\top}. \tag{12}$$

Starting from the definition $H = -\sum_i \pi_i \log \pi_i$ and using the chain rule,

$$\frac{\partial H}{\partial M(j \mid a)} = -\sum_i \frac{\partial \pi_i}{\partial M(j \mid a)}(1 + \log \pi_i) = -\sum_i \pi_i(\delta_{ij} - \pi_j)(1 + \log \pi_i). \tag{13}$$

Separating the term $i = j$ from the rest:

$$\frac{\partial H}{\partial M(j \mid a)} = -\pi_j(1 - \pi_j)(1 + \log \pi_j) + \pi_j \sum_{i \neq j} \pi_i(1 + \log \pi_i) \tag{14}$$

$$= \pi_j \Big[\sum_i \pi_i(1 + \log \pi_i) - (1 + \log \pi_j)\Big]. \tag{15}$$

Because $\sum_i \pi_i(1 + \log \pi_i) = 1 + \sum_i \pi_i \log \pi_i = 1 - H(\pi)$, we obtain

$$\frac{\partial H}{\partial M(j \mid a)} = \pi_j\big(1 - H(\pi) - 1 - \log \pi_j\big) = -\pi_j\big(\log \pi_j + H(\pi)\big), \tag{16}$$

which gives the stated component-wise form. Writing this for every j simultaneously yields the vector expression with the Hadamard product.

□

Lemma H.2 (Policy gradient for the conditional distribution)**.** *For an action $a \sim \pi_M(\cdot \mid \mathbf{s})$, sampled from a policy $\pi_M(\cdot \mid \mathbf{s})$, at state $\mathbf{s}$, the policy gradient is given by: $\nabla_{M_\mathbf{s}} \log \pi(a \mid \mathbf{s}) \cdot A(\mathbf{s}, a)$, where $A(\mathbf{s}, a)$ is the advantage of action a. The expression for the bth coordinate of the policy gradient can be written down in closed form as:*

$$\left[\nabla_{M_\mathbf{s}} \log \pi(a \mid \mathbf{s}) \cdot A(\mathbf{s}, a)\right]_b \;=\; \left(\mathbf{1}(b = a) - \pi(a \mid \mathbf{s})\right) \cdot A(\mathbf{s}, a),$$

where $\mathbf{1}(\cdot)$ is an indicator function.

Proof. Write $Z := \sum_c \exp M(c \mid \mathbf{s})$ and $\pi_b := \pi_M(b \mid \mathbf{s}) = \exp M(b \mid \mathbf{s})/Z$ for brevity. By definition

$$\log \pi_M(a \mid \mathbf{s}) \;=\; M(a \mid \mathbf{s}) \;-\; \log Z. \tag{17}$$

For any coordinate $b \in \bar{\mathcal{A}}$,

$$\frac{\partial}{\partial M(b \mid \mathbf{s})} \log \pi_M(a \mid \mathbf{s}) \;=\; \underbrace{\mathbf{1}(b = a)}_{\text{derivative of } M(a|\mathbf{s})} \;-\; \frac{1}{Z}\frac{\partial Z}{\partial M(b \mid \mathbf{s})}$$

$$=\; \mathbf{1}(b = a) - \frac{\exp M(b \mid \mathbf{s})}{Z} \;=\; \mathbf{1}(b = a) - \pi_b. \tag{18}$$

Multiplying every coordinate by the common scalar $A(\mathbf{s}, a)$ produces the stated expression for $g(\mathbf{s}, a; M)$. □

Theorem H.3 (Negative gradient increases $H(M; \mathbf{s})$ when $p(a^\star|\mathbf{s})$ is low)**.** *For any state $\mathbf{s}$, current parameters $M^{(i)}$, suppose the most likely action $\bar{a}$ is incorrect, i.e., $a^\star \neq \bar{a} =: \arg\max_b \pi_{M^{(i)}}(b \mid \mathbf{s})$, where the probability of sampling $\bar{a} \mid \mathbf{s}$ is $\pi_{\bar{a}}$, and the second most likely action has probability $\pi_{\bar{a}} - \varepsilon$. Then, for a small enough learning rate $\eta > 0$ s.t. with probability $\geq \pi_{\bar{a}}$, negative gradient produces $\pi^{(i+1)}$ with entropy $H(M^{(i+1)}; \mathbf{s}) > H(M^{(i)}; \mathbf{s})$. Additionally, there exists a universal constant $c > 0$ s.t., $H(M^{(i+1)}; \mathbf{s}) - H(M^{(i)}; \mathbf{s}) \geq c\eta \cdot K\varepsilon^2(1 - p_{\bar{a}})$ whenever $\pi_{\bar{a}} \geq \varepsilon + e^{-H(M^{(i)}; \mathbf{s})}$. In contrast, without negative gradient the entropy remains same with probability $1 - \pi(a^\star \mid \mathbf{s})$.*

Proof. For simplicity let us denote $\pi^{(i)} = \big(\pi_1, \ldots, \pi_{K+1}\big) \in \triangle(\bar{\mathcal{A}})$ be the conditional distribution produced by a bi-gram softmax column $\pi_{M^{(i)}}(\cdot \mid \mathbf{s})$, i.e., the proabability of sampling action a at state $\mathbf{s}$, with model parameters given by the current RL iterate $M^{(i)}$. Let us also denote,

$$\bar{a} \;=\; \arg\max_i \pi_i, \qquad H(M^{(i)}; \mathbf{s}) =: -\sum_{a \in \bar{\mathcal{A}}} \pi_a \cdot \log \pi_a,$$

where π_a is the probability of sampling action a at state $\mathbf{s}$. Given that the current policy π_M samples action $a \sim \pi^{(i)}(\cdot \mid \mathbf{s})$, the stochastic policy gradient that updates the parameter is given by:

$$M_{\mathbf{s}}^{(i+1)} = M_{\mathbf{s}}^{(i)} + \eta\, \nabla_{M_{\mathbf{s}}^{(i)}} \log(\pi^{(i)}(a \mid \mathbf{s})) \cdot A(\mathbf{s}, a), \tag{19}$$

where η is the learning rate. Note, that the policy parameters would only be updated for the row corresponding to the state $\mathbf{s}$. For simplicity, let us use the notation g for:

$$g =: \nabla_{M_{\mathbf{s}}^{(i)}} \log(\pi^{(i)}(a \mid \mathbf{s})) \cdot A(\mathbf{s}, a). \tag{20}$$

Then, $M_{\mathbf{s}}^{(i+1)} - M_{\mathbf{s}}^{(i)} = \eta \cdot g$. A second–order Taylor expansion of the concave function $H(M; \mathbf{s})$ gives, for some $\tilde{M}$ on the segment $[M^{(i)}, M^{(i+1)}]$:

$$\begin{aligned} H(M^{(i+1)}; \mathbf{s}) = H(M^{(i)}; \mathbf{s}) &+ \eta \cdot \langle \nabla_{M^{(i)}} H(M^{(i)}; \mathbf{s}), g \rangle \\ &+ \tfrac{\eta^2}{2} \cdot (g)^\top\, \nabla^2_{\tilde{M}_{\mathbf{s}}} H(\tilde{M}; \mathbf{s})\, (g)\,. \end{aligned} \tag{21}$$

Let the least eigenvalue of the Hessian of the conditional entropy (note that the entropy is a convae function) with respect to the logits be $\rho_{\tilde{M}_{\mathbf{s}}}$, and $|\rho_{\mathbf{s}}| < \infty$, the moment $\pi^{(i)}(a \mid \mathbf{s}) > 0$ for all actions $a \in \bar{\mathcal{A}}$. This condition is easily satisfied by any policy in our policy class, with finite values of the parameter matrix M. Thus, whenever $\langle g, \nabla_{M_{\mathbf{s}}^{(i)}} H(M^{(i)}; \mathbf{s}) \rangle > 0$ there exists a small enough learning rate η,

$$\eta \le \frac{2\langle g, \nabla_{M_{\mathbf{s}}^{(i)}} H(M^{(i)}; \mathbf{s}) \rangle}{\rho \|g\|_2^2}, \tag{22}$$

such that $H(M^{(i+1)}; \mathbf{s}) - H(M^{(i)}; \mathbf{s})$ is strictly positive. Thus, we can continue to reduce learning rate η such that we can ignore $\mathcal{O}(\eta^2)$ terms in Eq. 21, to get the bound:

$$H(M^{(i+1)}; \mathbf{s}) - H(M^{(i)}; \mathbf{s}) \ge \frac{\eta}{2} \cdot \langle \nabla_{M_{\mathbf{s}}^{(i)}} H(M^{(i)}; \mathbf{s}), \nabla_{M_{\mathbf{s}}^{(i)}} \log(\pi^{(i)}(a \mid \mathbf{s}) \cdot A(\mathbf{s}, a)) \rangle \tag{23}$$

Next, it remains to bound the right hand side of Eq. 23 with high probability over the sampling of the action a. For a single incorrect action draw $a \sim \pi$ we set $A(\mathbf{s}, a)$ to be -1 and for such an incorrect action we define the alignment scalar:

$$\mathcal{T}(a) \;=:\; -\Big\langle\, \nabla_{M_{\mathbf{s}}^{(i)}} \log \pi^{(i)}(a \mid \mathbf{s}) \cdot A(\mathbf{s}, a),\; \nabla_{M_{\mathbf{s}}^{(i)}} H(M^{(i)}; \mathbf{s}) \Big\rangle \tag{24}$$

Plugging in the derivation of $\nabla_{M^{(i)}} H(M^{(i)}; \mathbf{s})$ from Lemma H.1, we compute the closed form expression for $T(a_i)$ using the following definitions:

$$v_i =: \pi_i \big(H(M^{(i)}; \mathbf{s}) + \log \pi_i\big) \quad \text{and,} \quad \mu =: \sum_{a \in \bar{\mathcal{A}}} \pi_a v_a \tag{25}$$

Thus, one has $T(a)$ satisfy:

$$\mathcal{T}(a) = v_a - \mu \qquad \text{when,}\;\; a \in \bar{\mathcal{A}}, \;\; i \ne a^\star. \tag{26}$$

Note that v_i is an increasing function in π_i whenever $\pi_i > e^{-H(M^{(i)}; \mathbf{s})}$. Next, we note that $v_{\bar{a}} \ge 0$.

$$\pi_{\bar{a}} \ge \frac{1}{|\bar{\mathcal{A}}|} \quad \implies \quad \pi_{\bar{a}} \ge e^{-H(M^{(i)}; \mathbf{s})} \quad \text{since,}\;\; H(M^{(i)}; \mathbf{s}) \le \log |\bar{\mathcal{A}}| \quad \implies \quad v_{\bar{a}} \ge 0. \tag{27}$$

Finally, since $v(x) = xH(M^{(i)}; \mathbf{s}) + x \log x$ is convex in x:

$$v_{\bar{a}} \geq \sum_j \pi_j v_j \quad \implies \quad v_{\bar{a}} - \mu \geq 0 \tag{28}$$

The above two implications in Eq. 27 and Eq. 28, and the fact that $\bar{a} \neq a^\star$, together lead us to a deterministic lower bound on $T(\bar{a})$, implying that it is always positive:

$$\mathcal{T}(\bar{a}) \;\geq\; 0. \tag{29}$$

This completes the derivation for the first part of Theorem H.3, which does not assume anything about the conditional distribution $\pi^{((i))}(\cdot \mid \mathbf{s})$, directly yielding the following result.

Result (i): Under the conditional distribution $\pi^{(i)}(\cdot \mid \mathbf{s})$, whenever the most likely action $\bar{a} \neq a^\star$, then with probability at least $\pi_{\bar{a}}$, $T(a) \geq 0$, for $a \sim \pi^{(i)}(\cdot \mid \mathbf{s})$, and any policy π in our class of softmax policies. Finally, we plug this into Eq. 23 to conclude that the policy gradient update with probability $\pi_{\bar{a}}$ always increases entropy, for a small enough learning rate.

Next, we lower bound $T(\bar{a})$ when the second most likely action under the distribution satisfies an additional condition. For this, let us fix some $\varepsilon \geq 0$, such that for $q = \arg\max_{b \neq \bar{a}} \pi^{(i)}(b \mid \mathbf{s})$, we have $\pi_q = \pi_{\bar{a}} - \varepsilon$. Based on our alignment scalar $\mathcal{T}(\cdot)$, we define the function $g(x)$ as follows:

$$g(x) \;=\; x\big(H(M^{(i)}; \mathbf{s}) + \log x\big), \qquad 0 < x \leq 1, \tag{30}$$

where $H(M^{(i)}; \mathbf{s})$ is the conditional entropy we defined previously. Then, given the most probable action $\bar{a}$, and the runner up action q, the gap between $\mathcal{T}(\bar{a})$ can be lower bounded down as follows when $\pi_q \geq \exp(-H(M^{(i)}; \mathbf{s}) - 1)$:

$$\begin{aligned}\mathcal{T}(\bar{a}) &= g(\pi_{\bar{a}}) - \pi_{\bar{a}} \cdot g(\pi_{\bar{a}}) - \sum_{b \neq \bar{a}} \pi_g \cdot g(b) \\ &\geq (1 - \pi_{\bar{a}}) \cdot g(\pi_{\bar{a}}) - (1 - \pi_{\bar{a}}) \cdot g(q) = (1 - \pi_{\bar{a}}) \cdot (g(\pi_{\bar{a}}) - g(\pi_q)),\end{aligned} \tag{31}$$

where the second equality follows from the fact that $g(\pi_q) \geq g(b)$ for any $b \neq \bar{a}$ as soon as $\pi_q \geq \exp(-H(M^{(i)}; \mathbf{s}))$, which is implied by the condition on $\pi_{\bar{a}}, \varepsilon$ in Theorem H.3.

By the mean–value form of Taylor's theorem there exists a $\xi \in [\pi_q, \pi_{\bar{a}}]$ such that

$$g(\pi_{\bar{a}}) \;=\; g(q) \;+\; \varepsilon\, g'(q) \;+\; \frac{\varepsilon^2}{2}\, g''(\xi). \tag{32}$$

Because g is convex, $g''(\xi) = 1/\xi > 0$ and the linear term $\varepsilon g'(q)$ is non–negative. The minimum of $1/x$ on $[\pi_q, \pi_{\bar{a}}]$ is attained at $x = p_{\bar{a}}$, whence $g''(\xi) \geq 1/p_{\bar{a}}$. Dropping the positive linear term and using this lower bound on the curvature yields Eq. 33.

$$g(\pi_{\bar{a}}) - g(\pi_q) \;\geq\; \frac{\varepsilon^2}{2\,\pi_{\bar{a}}} \;\geq\; \frac{\varepsilon^2}{2} \cdot K, \tag{33}$$

since $\pi_{\bar{a}} \geq 1/K+1$. Plugging the above result into Eq. 31 we get the follow result.

Result (ii) Under the conditional distribution, $\pi^{(i)}(\cdot \mid \mathbf{s})$ whenever the most likely action $\bar{a} \neq a^\star$, and when the second most likely action q has probability $\pi_q \geq \exp\left(-H(M^{(i)}; \mathbf{s})\right)$, then with probability at least $\pi_{\bar{a}}$, $T(a) \geq c' \cdot K(\pi_{\bar{a}} - \pi_q)^2(1 - \pi_{\bar{a}})$, for $a \sim \pi^{(i)}(\cdot \mid \mathbf{s})$, and a universal constant $c' > 0$. Finally, we plug this into Eq. 23 to conclude that the policy gradient update with probability $\pi_{\bar{a}}$ always increases entropy by at least $c\eta \cdot K\varepsilon^2(1 - \pi_{\bar{a}})$, for a small enough learning rate.

Together, **Result (i, ii)** complete the proof of Theorem H.3. ☐

I. Broader Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. Our findings deepen our understanding of how to train large language models (LLMs) to reason more effectively under test-time compute constraints, which could influence the design of future AI systems. Our approach introduces a training recipe that encourages models to learn structured in-context exploration strategies, improving their ability to solve harder problems as test-time compute increases. While this has the potential to improve AI reasoning and decision-making across domains, it also raises questions around the unequal access to models that can leverage such capabilities, especially in settings with limited compute resources. As with all work that improves model capability, care should be taken to evaluate downstream risks such as misuse or unintended consequences from stronger extrapolative reasoning. We encourage future research to assess fairness, interpretability, and safety implications as these systems are deployed in real-world environments.

J. Note on Computational Resources Used for `e3`

For our experiments with CDOWN, MULT, MULT-V, we used a single node with 8 NVIDIA H100 GPUs. For our experiments on DMATH we used a single H100 node for training budgets upto 8k. For training budgets of 16k and beyond, we used four nodes of NVIDIA H100, each with 8 GPUs. Alternatively, we were able to run our experiments on CDOWN, MULT, MULT-V, and on DMATH (upto a training budget of 8k) using TPUs. In particular, we found that v5litepod-128 TPU, which has 32 nodes, 128 total chips (TPU configuration of 32×4) is sufficient to run our experiments. Our experiments on GPUs used the open-source LLM post-training framework from Sheng et al. (2024) that uses FSDP sharding to parallelize data and split model parameters across the multiple cards.

K. Example Traces

Example 1: Chained Asymmetries in Countdown

```
<think> Let me analyze this step by step:
1.  First, let's try to get close to 466 with simple operations:  65 + 56 = 121 (too
small) 65 - 56 = 9 (too small) 65 * 56 = way too large 56 * 65 = way too large
2.  Let's try working backwards from 466:  466 is much larger than any of the given
numbers, so multiplication will likely be needed.  We need to get from 37 to 466,
which means we need to multiply by 12.5.  But we don't have 12.5, so we'll need to
create it through division.
3.  Let's try:  37 * 14 = 518 (too large) 37 - 14 = 23 (too small) 37 + 14 = 51 (too
small)
4.  Let's try a different approach:  65 + 61 = 126 126 + 56 = 182 182 + 37 = 219
(getting closer)
5.  Let's try:  65 + 61 = 126 126 + 56 = 182 182 + 37 = 219 219 + 14 = 233 (still
too small)
6.  Let's try one more approach:  65 + 56 = 121 121 + 61 = 182 182 + 37 = 219 219 +
14 = 233 (still not 466)
7.  Let's try:  65 + 56 = 121 121 + 61 = 182 182 + 37 = 219 219 + 14 = 233 (still
not 466)
8.  One more attempt:  65 + 61 = 126 126 + 37 = 163 163 + 56 = 219 219 + 14 = 233
(still not 466)
9.  Final try:  65 + 61 = 126 126 + 56 = 182 182 + 37 = 219 219 - 14 = 205 (not
quite there)
10.  Let's try:  65 + 61 = 126 126 + 56 = 182 182 + 37 = 219 219 + 14 = 233 (still
not 466)
11.  New approach:  65 + 61 = 126 126 + 56 = 182 182 - 37 = 145 145 + 14 = 159 (not
quite there)
12.  Final attempt:  65 + 61 = 126 126 + 56 = 182 182 + 37 = 219 219 - 14 = 205 (not
quite there)
13.  Let's try:  65 + 56 = 121 121 + 61 = 182 182 + 37 = 219 219 - 14 = 205 (not
quite there)
14.  One last try:  65 + 37 = 102 102 + 61 = 163 163 + 56 = 219 219 + 14 = 233 (not
quite there)
15.  Final attempt:  56 + 14 = 70 70 + 65 = 135 135 + 37 = 172 172 + 61 = 233 (not
quite there)
16.  New approach:  37 * 14 = 518 518 - 65 = 453 453 + 61 = 514 514 - 56 = 458 (not
quite there)
17.  Final try:  61 * 14 = 854 854 - 65 = 789 789 - 56 = 733 733 - 37 = 696 (not
quite there)
```

```
18. Let's try: 37 * 14 = 518 518 - 65 = 453 453 + 61 = 514 514 - 56 = 458 (not
quite there)
19. New approach: 56 * 37 = 2,052 2,052 / 65 approx. 31.5 (not helpful)
20. Final attempt: 65 * 14 = 910 910 - 61 = 849 849 - 56 = 793 793 + 37 = 830 (not
quite there)
21. New approach: 61 * 37 = 2,237 2,237 - 65 = 1,172 1,172 - 56 = 1,116 1,116 + 14
= 1,130 (not quite there)
22. New approach: 37 * 14 = 518 518 + 65 = 583 583 - 61 = 522 522 + 56 = 578 (not
quite there)
23. Final attempt: 65 * 14 = 910 910 - 61 = 849 849 + 56 = 905 905 - 37 = 868 (not
quite there)
24. New approach: 37 * 14 = 518 518 - 56 = 462 462 + 65 = 527 527 - 61 = 466 (This
works!) </think> <answer> (37 * 14 - 56 + 65) - 61 </answer>
```

Example 2: Multiplication with verification

```
Okay, so I need to multiply 347 by 583. Hmm, let me think about how to approach
this. I guess I can use the standard multiplication algorithm where you break it
down by each digit. Let me start by writing both numbers down:
347
× 583
-----
First, I'll multiply 347 by each digit of 583 starting from the rightmost
digit, which is 3 in this case. Then move to the next digit (8) and finally the
leftmost digit (5), remembering to add the appropriate zeros for each subsequent
multiplication. After that, I'll sum all the results together. Let me start with
the rightmost digit.
Starting with the units place: 3 × 347. Let me compute that:
3 × 7 = 21. I write down 1 and carry over 2.
3 × 4 = 12, plus the carried over 2 makes 14. Write down 4 and carry over 1.
3 × 3 = 9, plus the carried over 1 makes 10. Write down 10.
So, 3 × 347 = 1041. Let me check that again:
3 × 7 = 21 (correct), 3 × 4 = 12 + 2 = 14 (correct), 3 × 3 = 9 + 1 = 10 (correct).
So that's 1041. Alright.
Check: 3 × 347 = 1041. Correct.
Next, the tens place. The digit here is 8, but since it's in the tens place, it's
actually 80. So, we'll multiply 347 by 80. To make it easier, I can multiply 347
by 8 and then add a zero at the end.
8 × 347:
8 × 7 = 56. Write down 6, carry over 5.
8 × 4 = 32, plus the carried over 5 makes 37. Write down 7, carry over 3.
8 × 3 = 24, plus the carried over 3 makes 27. Write down 27.
So, 8 × 347 = 2776. Then add the zero for the tens place, making it 27760. Let me
verify:
8 × 347. 8×300=2400, 8×40=320, 8×7=56. 2400+320=2720, 2720+56=2776. Then adding
the zero gives 27760. Correct.
Check: 8 × 347 = 2776, so 2776 × 10 = 27760. Correct.
Now, the hundreds place. The digit here is 5, but it's in the hundreds place, so
it's 500. So, multiply 347 by 500. Again, multiply 347 by 5 and add two zeros.
5 × 347:
5 × 7 = 35. Write down 5, carry over 3.
5 × 4 = 20, plus the carried over 3 makes 23. Write down 3, carry over 2.
5 × 3 = 15, plus the carried over 2 makes 17. Write down 17.
So, 5 × 347 = 1735. Then add two zeros, making it 173500. Let me verify:
5 × 347. 5×300=1500, 5×40=200, 5×7=35. 1500+200=1700, 1700+35=1735. Then adding
two zeros gives 173500. Correct.
Check: 5 × 347 = 1735, so 1735 × 100 = 173500. Correct.
Now, add all three results together: 1041 + 27760 + 173500.
Let me do this step by step. First, add 1041 and 27760:
1041 + 27760. Let me line them up:
1041
+27760
```

```
-----
28801
Wait, 1041 + 27760: 1000 + 27760 = 28760, then +41 = 28801. Let me check with
another method:
20000 + 7000 = 27000, 700 + 60 = 760, 27000 + 760 = 27760. Then 27760 + 1000 =
28760, +41 = 28801. Correct.
Check: 1041 + 27760 = 28801. Correct.
Now add this result to 173500:
28801 + 173500. Let me line them up:
28801
+173500
------
202301
Wait, 28000 + 173500 = 201500, then 801 + 0 = 801, so total 201500 + 801 = 202301.
Let me verify another way:
28801 + 173500. Break down 173500 into 170000 + 3500. Then 28801 + 170000 = 198801.
Then 198801 + 3500 = 202301. Correct.
Check: 28801 + 173500 = 202301. Correct.
So, putting all together, the final result is 202,301. Let me just cross-verify
using another method to ensure I didn't make any mistakes in the addition.
</think>
The product of 347 and 583 is [202301].
```

Example 3: Over-exploration in Countdown (after training only on easy problems with 3 candidates, each attempt contains at most 3 equations when 5 is needed)

```
<think> Let me try to find a way to get to 66. First, let me look for numbers close
to 66 or factors/multiples that might be useful.
66 is close to 70, which is one of our numbers. Maybe we can suBtract or add to get
to 66. 70 - 4 = 66, so maybe we can create 4 somewhere.
Looking at the other numbers: 77 - 70 = 7 97 - 73 = 24 (interesting, we already
have 24) 24 + 24 = 48 (not helpful) 24 * 24 = 576 (too large)
Let me try a different approach: 77 - 70 = 7 97 - 73 = 24 24 + 7 = 31 (not helpful)
New approach: 97 - 73 = 24 24 + 70 = 94 (too far)
Let's try: 77 - 70 = 7 7 * 24 = 168 (too far)
One more attempt: 70 + 24 = 94 94 - 97 = -3 -3 * 77 = -231 (too far)
Final attempt: 97 - 24 = 73 73 - 70 = 3 3 * 77 = 231 (too far)
Let's try: 70 + 24 = 94 94 - 77 = 17 17 + 65 = 82 (too far)
Wait: 24 * 73 = 1748 1748 - 77 = 1671 (too far)
Last try: 77 + 24 = 101 101 - 70 = 31 31 - 97 = -66 (not what we want)
Let's try: 97 - 73 = 24 24 + 70 = 94 94 - 24 = 70 (not helpful)
Final attempt: 70 * 24 = 1680 1680 / 77 = 21.9 (not valid)
Last try: 73 - 65 = 8 8 * 24 = 192 192 / 97 = 1.98 (not valid)
I need to keep trying different combinations. What if: 24 / 3 = 8 So: 97 - 73 =
24 24 / 24 = 1 (not helpful)
Final attempt: 77 - 97 = -20 -20 * 24 = -480 -480 + 70 = -410 (too far)
Last try: 70 * 24 = 1680 1680 / 77 = 21.9
So the most promising path seems to be: 97 - 73 = 24 24 + 70 = 94 94 - 77 = 17 This
gets us too far from 66.
Let's try: 77 - 70 = 7 7 * 24 = 168 168 - 97 = 171 (too far)
Final attempt: 97 + 70 = 167 167 - 24 = 143 143 / 73 = 1.97
Wait: 24 * 65 = 1560 1560 / 77 = 20.26
Last try: 77 - 65 = 12 12 * 24 = 288 288 / 97 = 3
I need to keep trying different combinations. What if: 97 - 70 = 27 27 / 24 =
1.125
Wait: 24 + 97 = 121 121 - 73 = 48 48 + 77 = 125 (too far)
Last attempt: 70 + 24 = 94 94 - 77 = 17 17 + 65 = 82
One final try: 24 * 65 = 1560 1560 / 97 = 16.06
Final solution: 97 - 73 = 24 24 * 65 = 1560 1560 / 77 = 20.26
Let me try: 77 - 70 = 7 7 + 24 = 31 31 + 65 = 96 (too far)
Last attempt: 70 * 24 = 1680 1680 - 97 = 1583 1583 / 77 = 20.59
I need to keep trying. What if: 97 + 24 = 121 121 - 73 = 48 48 + 65 = 113
```

```
Wait:  65 - 24 = 41 41 + 77 = 118 118 - 70 = 48
Final try:  24 * 65 = 1560 1560 + 70 = 1630 1630 / 77 = 21.16
One last attempt:  77 * 24 = 1848 1848 / 65 = 28.46
Last try:  70 * 24 =
```

L. Conclusion and Limitations

We introduce in-context exploration as a core capability to enable extrapolation of test compute in reasoning models, beyond the token budget they are trained for. Our proposed recipe, `e3`, leverages (1) asymmetries in the base model, (2) negative gradients in RL, and (3) a coupled curriculum over data and token budget to train a model that can perform in-context exploration. Applied to Qwen3-1.7B model, our method achieves state-of-the-art performance on the AIME/HMMT'25 benchmarks, with particularly strong gains in the extrapolation regime. Despite these, our study is limited in terms of model scale and domain. Future work should explore how `e3` generalizes to larger model scales and other reasoning domains. For a brief statement on broader statement please see App. I.